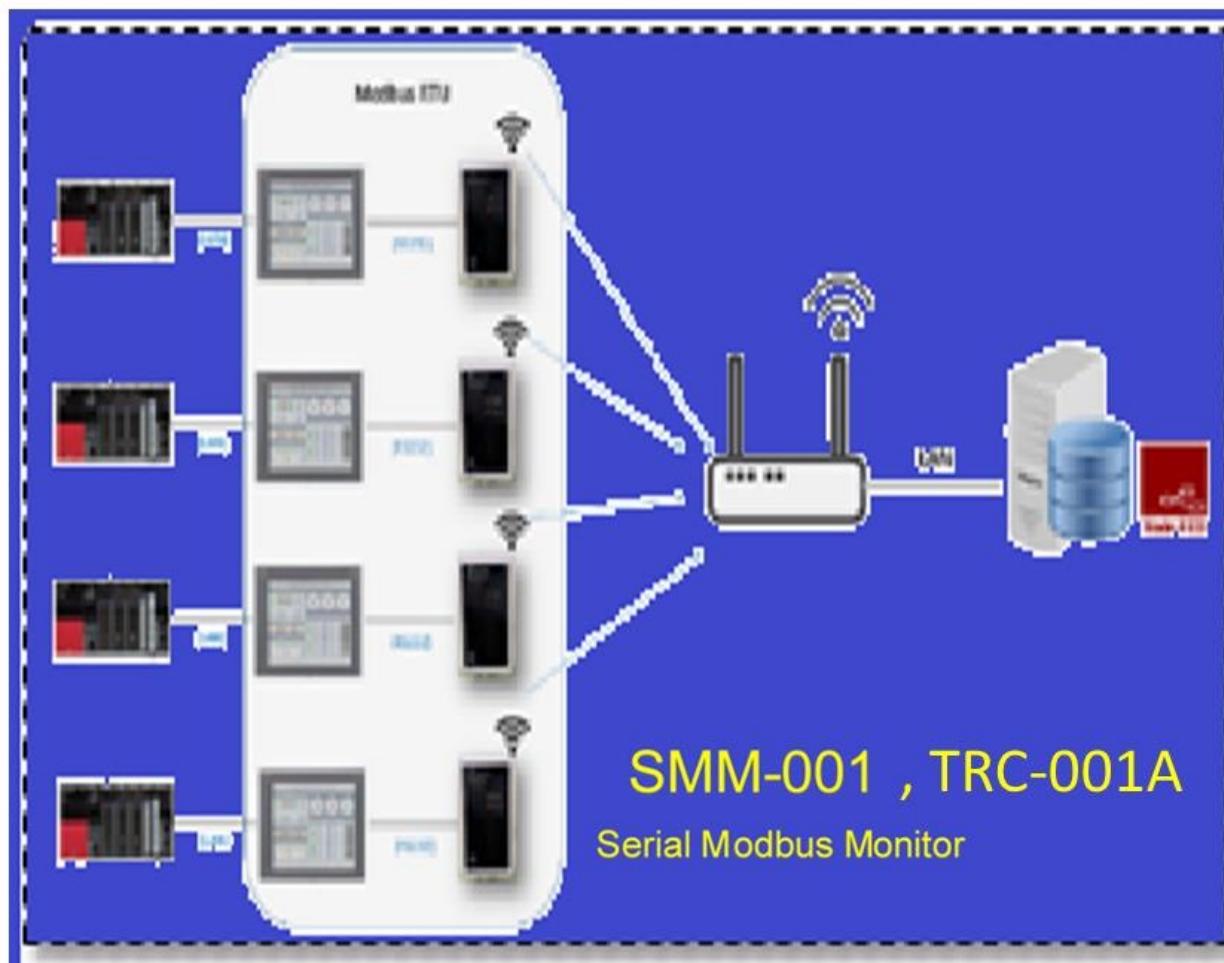


## BASIC Knowledge for IOT Development

เอกสารประกอบการอบรมหลักสูตร IOT Development ระดับเริ่มต้น



- การใช้งาน MIC SMART รุ่น TRC-001A
- การใช้งาน Software Arduino IDE, GX WORKS 3, GT Designer3, Node-Red, SQL Server สำหรับออกแบบระบบ IOT เป็นต้น

# CONTENTS

---

## CHAPTER 1: OVERVIEW MIC SMART

---

1.1	MIC SMART คืออะไรและทำอะไรได้บ้าง?	1
1.2	ข้อดีของ MIC SMART	2
1.3	การทำงานของ MIC SMART รุ่น TRC-001A	2

## CHAPTER 2: MIC SMART CONNECTIONS

---

2.1	USB Connection to Computer	4
2.2	Rs232 Connection to Computer	5
2.3	Serial Modbus RTU GOT2000	6
2.4	Serial Modbus RTU PLC	12

## CHAPTER 3: FEATURES OF Arduino IDE and Installation

---

3.1	FEATURES OF Arduino IDE	17
3.2	Installation Arduino IDE	19
3.3	ESP32S2 CUCUMBER Hardware	21
3.4	Installation Libraries	26
3.5	Create New Project	30
3.6	C++ language structure used in the Arduino IDE	31

# CONTENTS

---

## CHAPTER 4: INTRODUCTION TO MIC SMART

4.1 การสั่งงานหลอด LED	32
4.2 การสร้าง Function.	33
4.3 RGB LED	33
4.4 Buzzer	34
4.5 Serial Monitor	36
4.6 ชนิดของข้อมูลและการใช้ตัวแปร	37
4.7 ตัวแปรอาร์เรย์ (Array)	38
4.8 เงื่อนไข Condition	40
4.9 Digital Input	45
4.10 LCD	48
4.11 Sensor SHT4X	49
4.12 Watch Dog Timer	51
4.13 Modbus Master	52
4.14 Modbus Slave	55
4.15 OTA Static IP	56
4.16 PubSubClient	58

## CHAPTER 5: MQTT Protocol

---

5.1 Node-red Structure	62
5.2 วิธีการ Publish และ Subscribe ข้อมูลจาก MIC SMART ไปที่ Node-red	65
5.3 วิธีการรับค่าจาก Modbus Slave ส่ง Data ขึ้น Node-red	67
5.4 วิธีการนำค่าที่ได้เข้าไปใน SQL Server	73

## CHAPTER 1: OVERVIEW MIC SMART

### 1.1 MIC SMART คืออะไรและทำอะไรได้บ้าง?



MIC SMART รุ่น TRC-001A

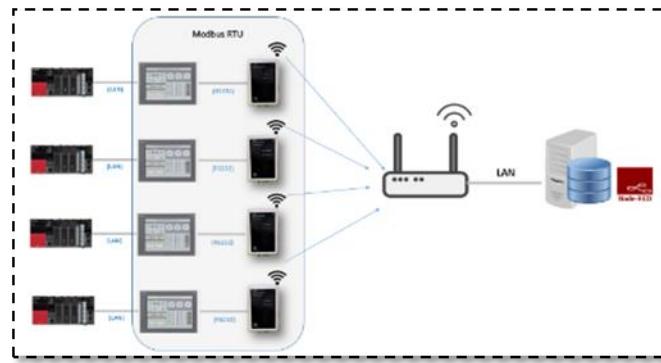
MIC SMART คืออุปกรณ์ IOT Gateway (Internet of Things Gateway) ที่เชื่อมต่อระหว่าง อุปกรณ์อัจฉริยะ เช่น PLC,HMI เข้ากับอินเทอร์เน็ต (Internet) โดยใช้โปรโตคอลต่าง ๆ เช่น Wi-Fi, Bluetooth, RS232 เป็นต้น แล้วส่งข้อมูลจากอุปกรณ์ IOT ไปยัง Server หรือบริการคลาวด์ (Cloud Service) เพื่อให้ผู้ใช้สามารถเข้าถึงและจัดการกับข้อมูลนั้นได้ อีกทั้ง MIC SMART ยังสามารถ จัดการการเชื่อมต่อของอุปกรณ์ IOT ต่าง ๆ ในระบบได้อย่างมีประสิทธิภาพ อาทิเช่น การส่งข้อมูล จากอุปกรณ์ต่าง ๆ ไปยังศูนย์กลางของระบบ (Central Hub) เพื่อจัดการข้อมูลและควบคุมการทำงาน ของอุปกรณ์ IOT ให้มีประสิทธิภาพมากยิ่งขึ้น

### 1.2 ข้อดีของ MIC SMART

- ผู้ใช้งานสามารถนำไปใช้งานได้หลากหลายโดยการพัฒนาโปรแกรมขึ้นมาเองได้
- สามารถต่อกับอุปกรณ์ควบคุมต่างๆ ที่มีการสื่อสาร Modbus ,Serial port ต่างๆได้

### 1.3 การทำงานของ MIC SMART รุ่น SMM-001A

MIC SMART จะต้องเชื่อมต่อกับอุปกรณ์ควบคุมเครื่องจักรต่างๆ เช่น PLC HMI และ Controller ที่มี port RS232 จากนั้น MIC SMART จะรับค่าจากอุปกรณ์นั้นๆ และส่งค่าข้อมูลขึ้น Server หรือบริการคลาวด์ (Cloud Service) ผ่านทางระบบ Wi-Fi



ตัวอย่างการเชื่อมต่อ HMI และระบบ Network

### รูปถ่ายหน้าจอของ MIC SMART รุ่น TRC-001A



หน้าจอของ MIC SMART จะมีหลอด LED อยู่ 3 หลอด ดังนี้

1. Connection ทำหน้าที่แสดงสถานะการติดต่อระบบ Wi-Fi และ Server
2. RUN ทำหน้าที่แสดงสถานะในการส่งข้อมูลหรือการทำงานต่างๆ

3. Result ทำหน้าที่แสดงสถานะการทำงาน ได้หลากหลาย เป็นต้น ได้หลายสี

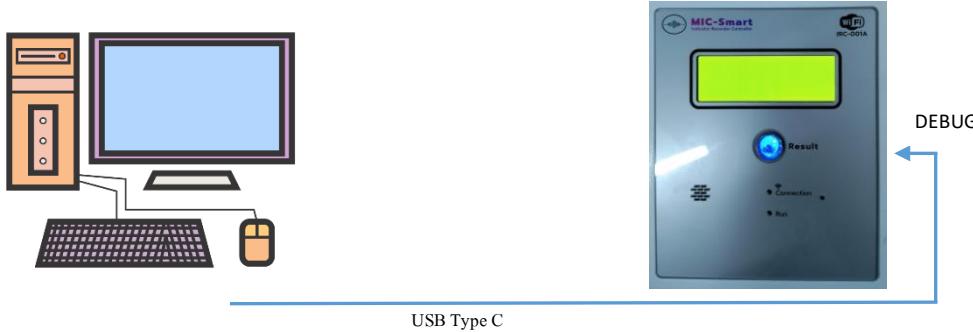


ด้านข้างอุปกรณ์ จะประกอบด้วย 5 พอร์ต

1. USB OTG ใช้สำหรับเชื่อมต่อกับอุปกรณ์ USB อื่น ๆ เพื่อรับส่งข้อมูล โดยมีโปรโตคอล USB 2.0 ที่รองรับตัวควบคุมเครือข่าย (network controller) และตัวควบคุมอุปกรณ์เก็บข้อมูล (mass storage device controller) อีกด้วย นอกจากนี้ ESP32-S2 ยังสามารถเป็นเครื่องเซิร์ฟเวอร์ USB และเป็นเครื่องมือสำหรับเชื่อมต่อกับอุปกรณ์ USB อื่น ๆ ได้อีกด้วย
2. USB DEBUG ใช้สำหรับ Upload Program และทดสอบตรวจสอบการทำงานของโปรแกรมได้
3. RS232 Port ใช้สำหรับสื่อสารอุปกรณ์ภายนอก
4. GPIO ใช้สำหรับรับสัญญาณ Digital Input และ Digital Output

## CHAPTER 2: MIC SMART CONNECTIONS

### 2.1 USB Connection to Computer



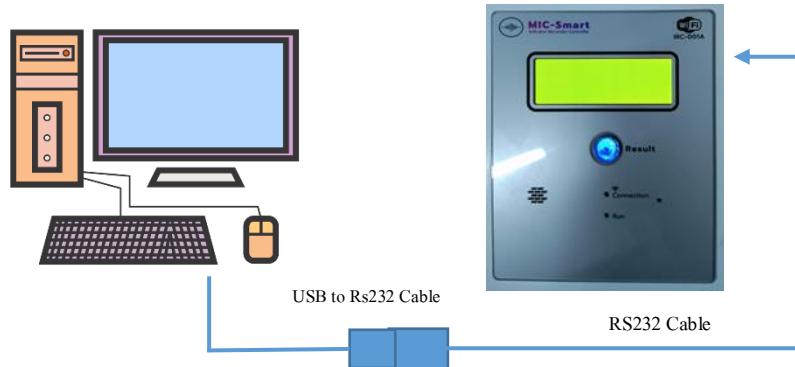
#### USB Connection to Computer

การใช้ USB Debug กับ MIC-SMART เป็นวิธีการตรวจสอบและแก้ไขข้อผิดพลาดในโปรแกรมที่พัฒนาขึ้นบน ESP32-S2 ได้ง่ายและมีประสิทธิภาพมากขึ้น เนื่องจาก USB Debug ช่วยให้ผู้พัฒนาสามารถเชื่อมต่อกับ MIC-SMART เพื่อดูข้อมูลต่าง ๆ เกี่ยวกับการทำงานของโปรแกรม และสามารถรับส่งข้อมูลได้แบบ real-time ดังนี้

- ตรวจสอบสถานะการทำงานของโปรแกรม ผู้พัฒนาสามารถตรวจสอบสถานะของโปรแกรมที่ถูกโหลดลงใน MIC-SMART ได้ รวมถึงดูข้อมูลต่างๆ เช่น ค่าตัวแปร เป็นต้น
- แก้ไขข้อผิดพลาด ผู้พัฒนาสามารถแก้ไขข้อผิดพลาดที่พบในโปรแกรมได้โดยตรง
- ทดสอบโปรแกรม ผู้พัฒนาสามารถทดสอบโปรแกรมบน MIC-SMART ได้และตรวจสอบค่าต่าง ๆ ของโปรแกรมได้ทันที
- อัพโหลดโปรแกรม ผู้พัฒนาสามารถอัพโหลดโปรแกรมบน MIC-SMART ได้โดยตรงผ่าน USB Debug

USB Debug

## 2.2 RS232 Connection to Computer



RS232 Connection to Computer

USB to RS232 Cable เป็นสายสื่อสารที่ใช้เชื่อมต่อระหว่างอุปกรณ์พีซี (PC) ที่มีพอร์ต USB กับอุปกรณ์ที่มีพอร์ต RS232 (เช่น อุปกรณ์ควบคุมอุณหภูมิ เครื่องวัดความดัน หรืออุปกรณ์อื่นๆที่ใช้งานด้วย RS232) เพื่อให้สามารถรับส่งข้อมูลระหว่างอุปกรณ์สองตัวได้

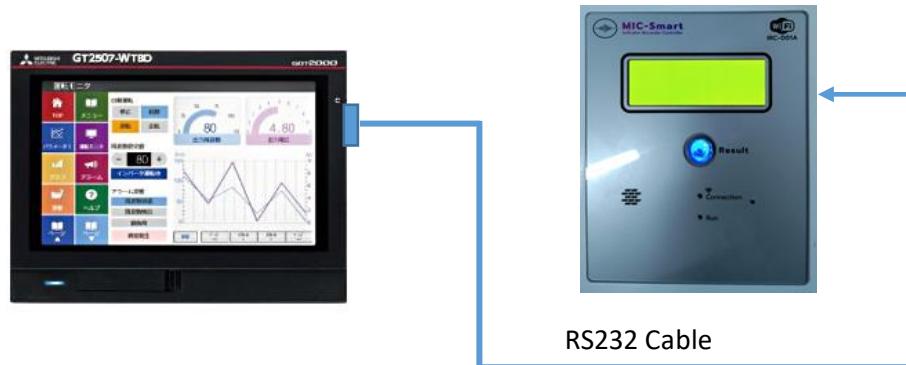
สาย USB to RS232 มักถูกนำมาใช้ในการเชื่อมต่อกับอุปกรณ์เครื่องมือวัดและควบคุมทางอุตสาหกรรม เนื่องจากอุปกรณ์เหล่านี้มักมีการใช้งานแบบ RS232 เป็นพอร์ตการสื่อสารข้อมูลโดยตรง โดยสาย USB to RS232 จะช่วยให้เราสามารถเชื่อมต่อกับอุปกรณ์เหล่านี้ผ่านพอร์ต USB ของเครื่องคอมพิวเตอร์ได้โดยไม่ต้องมีการเปลี่ยนแปลงพอร์ตหรือการ์ดขยายพอร์ตใดๆ อีกทั้งยังช่วยให้การเชื่อมต่อระหว่างเครื่องคอมพิวเตอร์กับอุปกรณ์ RS232 เป็นไปได้อย่างง่ายดายและสะดวกมากขึ้น



USB to RS232 Cable

สาย USB to RS232 จะต่อเข้ากับสาย RS232 ของชุด MIC-SMART เมื่อนำมาใช้งาน สำหรับทดสอบอุปกรณ์ MIC-SMART ต้องสารกับคอมพิวเตอร์ผ่านทาง RS232

### 2.3 Serial Modbus RTU GOT2000



Serial Modbus RTU GOT2000

Modbus RTU เป็นโปรโตคอลสื่อสารที่ใช้ในการติดต่อสื่อสารกันระหว่างอุปกรณ์อิเล็กทรอนิกส์ เช่น คอนโทรลเลอร์ PLC, ชุดควบคุมอุณหภูมิ เป็นต้น โดยโปรโตคอล Modbus RTU ใช้เทคนิคสื่อสารแบบ serial ซึ่งมีการส่งข้อมูลด้วยการส่งบิตที่ต่อเนื่องกัน และใช้ระบบส่งข้อมูลแบบ Half-Duplex คือสามารถส่งข้อมูลได้ทั้งขาส่งและขารับ แต่ไม่สามารถส่งข้อมูลได้พร้อมกันทั้งสองขา การสื่อสาร Modbus RTU จะมีการส่งข้อมูลแบบฟอร์แมตข้อมูล โดยมีโครงสร้างของข้อมูลดังนี้

1. ที่อยู่ Slave หรือ Device Address (1 byte)
2. รหัสฟังก์ชัน Function Code (1 byte)
3. ข้อมูล Data (n byte)
4. ตัวตรวจสอบ Checksum (2 byte)

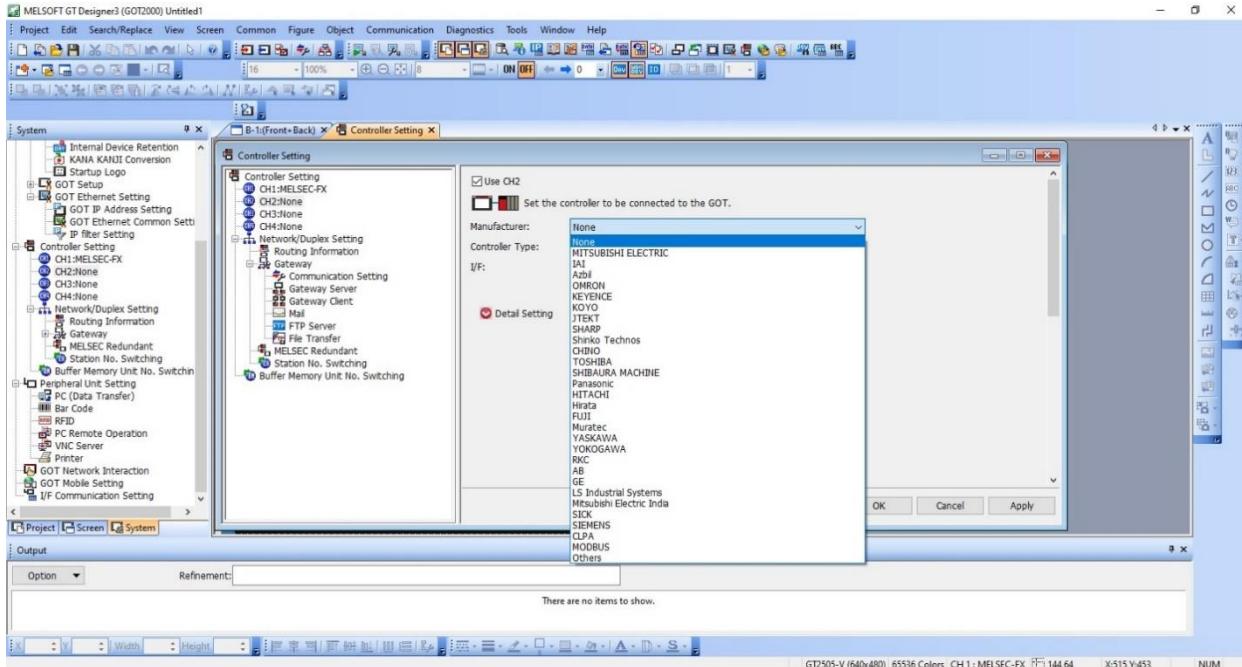
โครงสร้างของข้อมูลนี้จะแตกต่างกัน ไปป็นอยู่กับรหัสฟังก์ชันที่ถูกใช้งาน โดย Modbus RTU มีรหัสฟังก์ลักษณะ 4 รหัสฟังก์ชัน ดังนี้

1. Read Coil Status (Function Code 01) - ใช้อ่านสถานะของ Coil จาก Slave Device
2. Read Input Status (Function Code 02) - ใช้อ่านสถานะของ Input จาก Slave Device
3. Read Holding Registers (Function Code 03) - ใช้อ่านค่าของ Holding Registers จาก Slave Device
4. Read Input Registers (Function Code 04) - ใช้อ่านค่าของ Input Registers จาก Slave Device

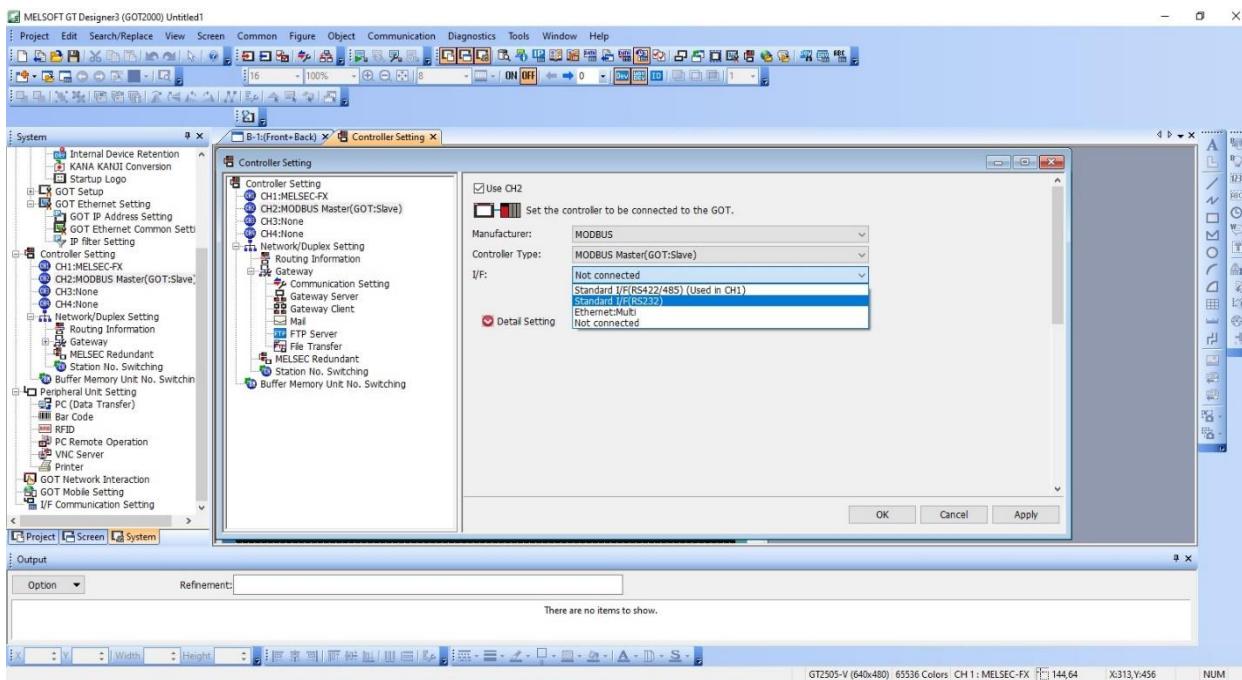
นอกจากนี้ยังมีฟังก์ชันอื่น ๆ เพิ่มเติมที่ใช้สำหรับการเขียนข้อมูล หรือการควบคุมอุปกรณ์ เช่น Write Single Coil (Function Code 05), Write Single Register (Function Code 06), Write Multiple Coils (Function Code 15), และ Write Multiple Registers (Function Code 16) ซึ่งเป็นฟังก์ชันที่ใช้สำหรับการเขียนข้อมูลลงใน Slave Device อีกด้วย

### ตัวอย่างวิธีเซตพารามิเตอร์ใน GT Designer

เมื่อสร้างโปรเจ็คใหม่เรียบร้อยแล้ว ให้ไปที่ System -->Controller setting-->CH2 จากนั้นคลิกถูก Use CH2 ไปที่ Manufacturer: MODBUS

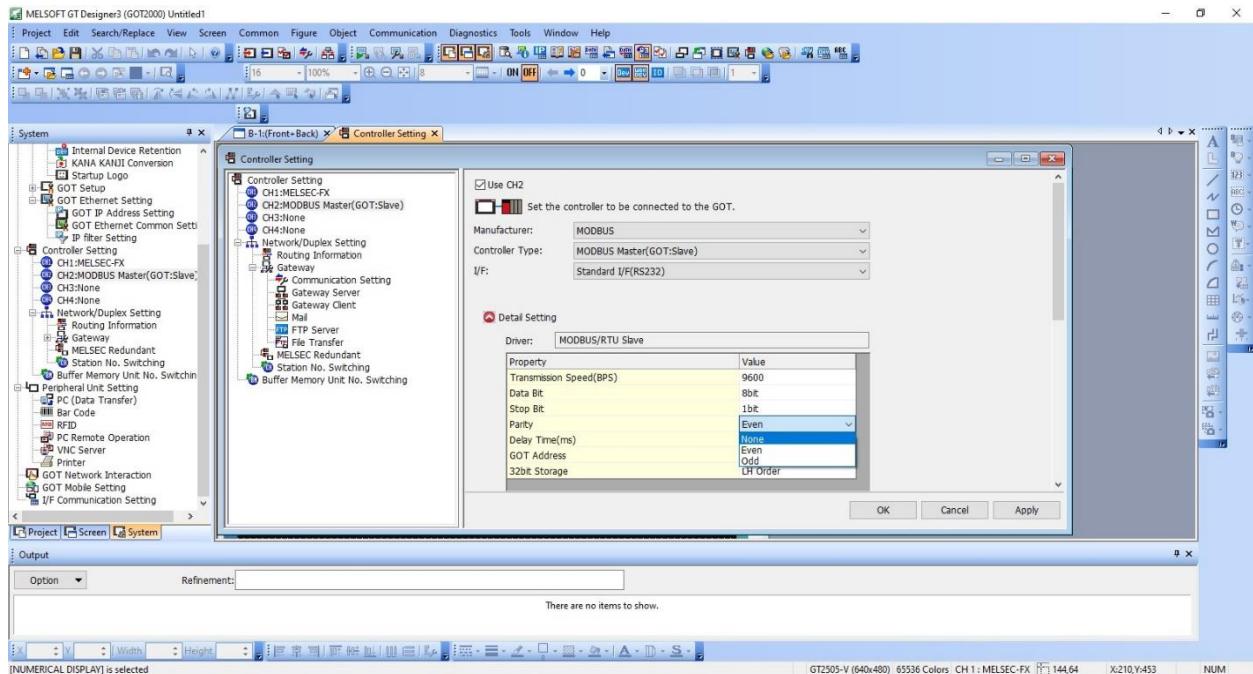


เลือก Controller Type : MODBUS MASTER(GOT:Slave) และ I/F: Standard I/F(RS232)



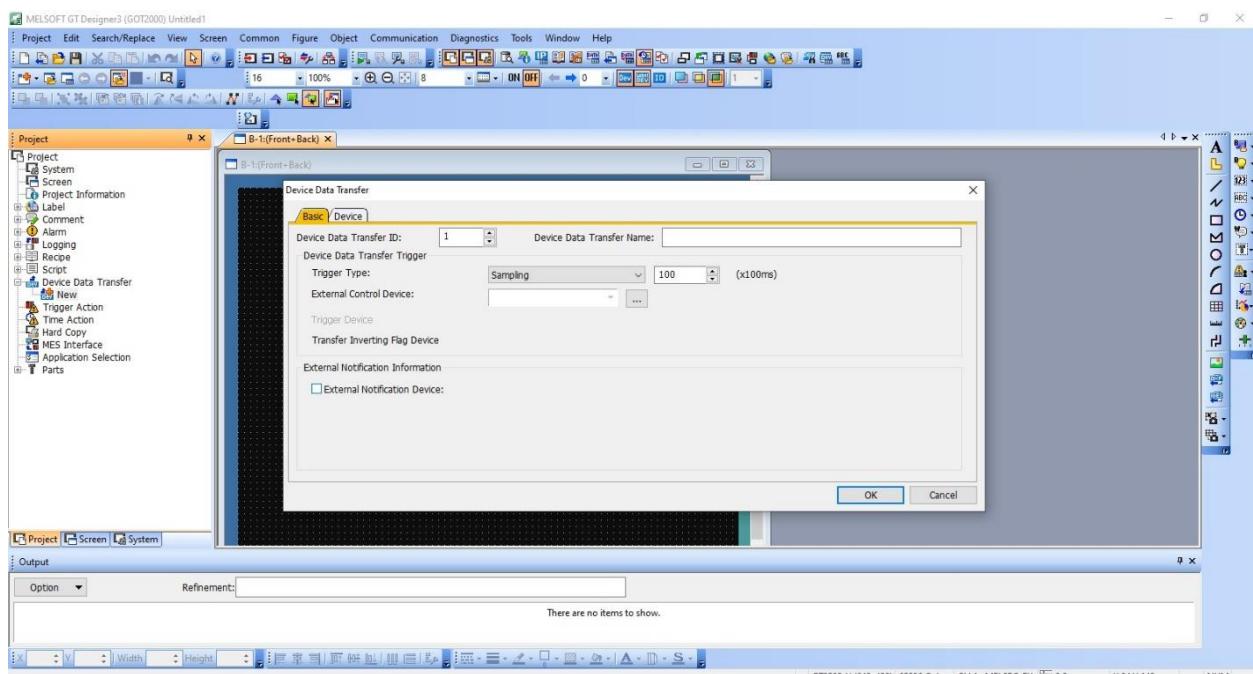
คลิกที่ Detail Setting กำหนดค่า Transmission Speed(BPS) : 9600 และ Parity : None

จากนั้นกด OK

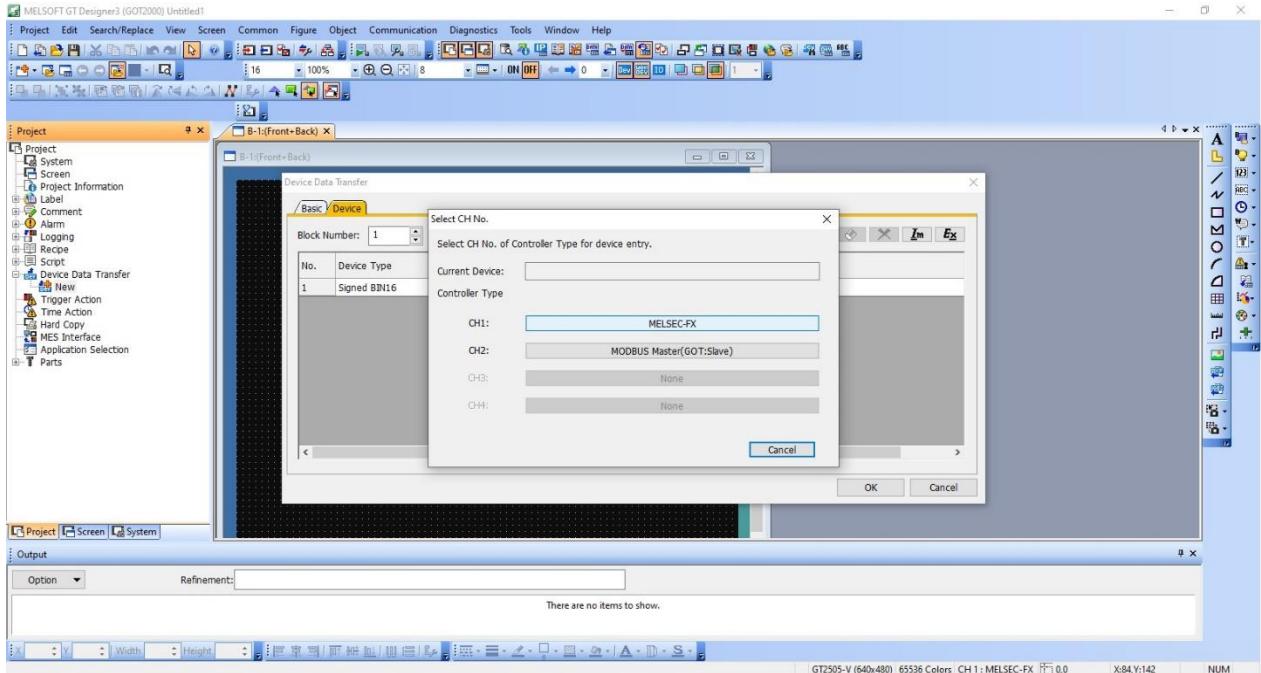


คลิก Project เลือก Device Data Transfer → New จากนั้นคลิก Basic → Trigger Type:

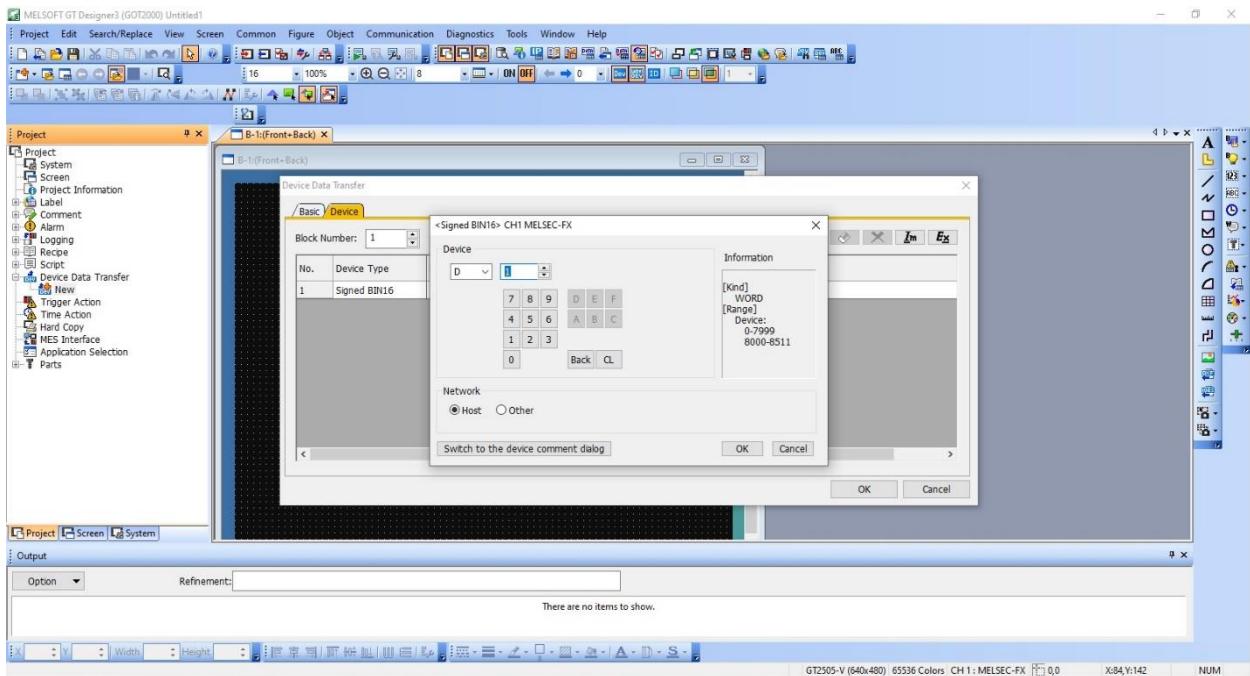
เลือกเป็น Sampling → 10x100ms (1 sec)



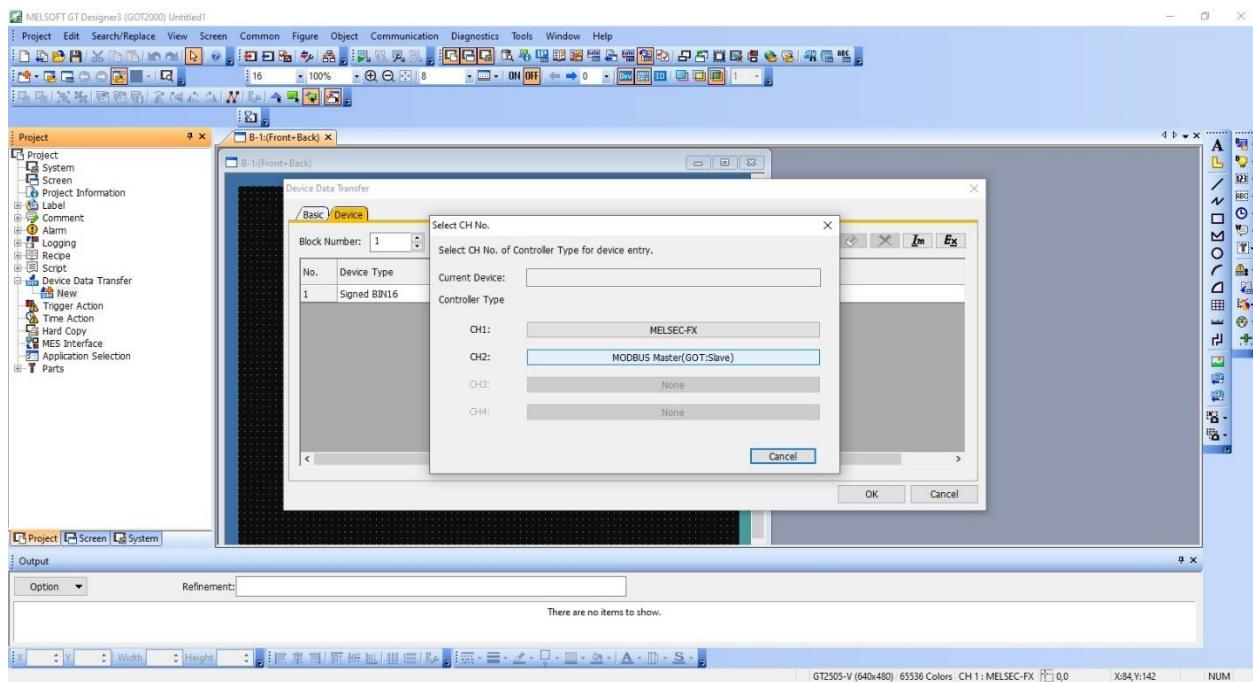
คลิก Device เลือก Souce Device เป็น CH1: MELSEC-XX



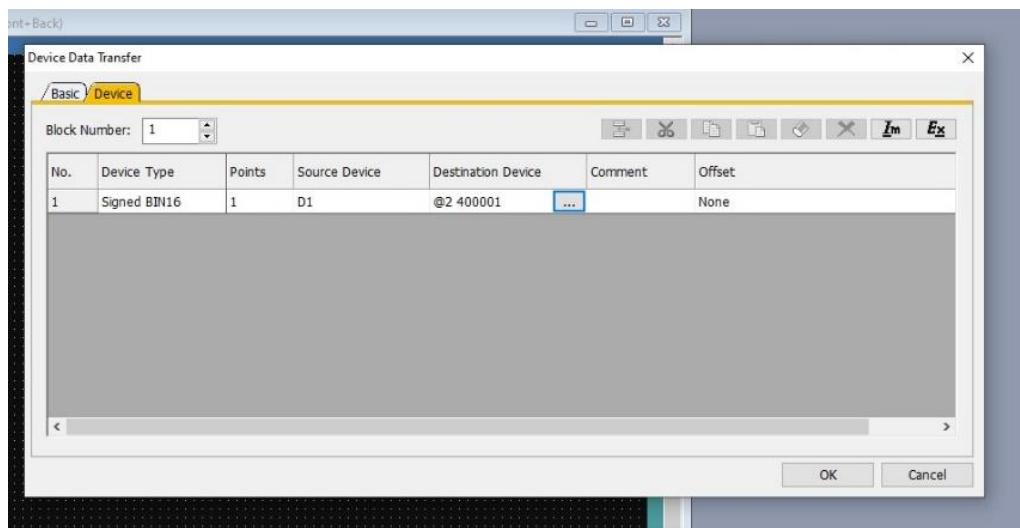
เลือก Data Register ที่ต้องการจึงค่าในที่นี่เลือก D1 แล้วกด OK



เลือก Destination Device เป็น CH:2 MODBUS Master(GOT:Slave)



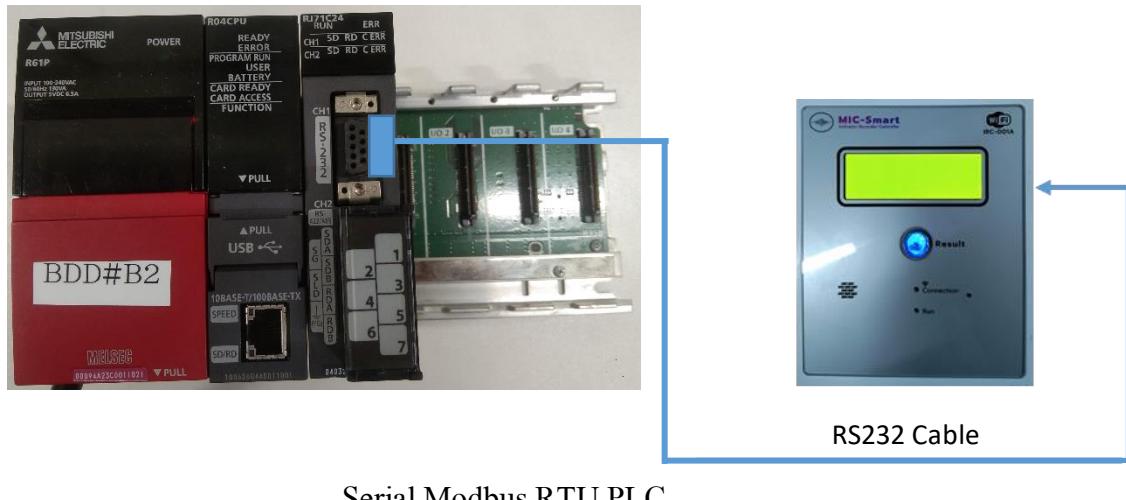
กำหนด Holding Registers เป็น 400001 ( Holding Registers ตัวแรก) จะได้ดังรูปแล้วกด OK



ตัวอย่างในการตั้งค่า GOT กำหนดให้ Holding Registers ลำดับที่ 400001 รับค่าจาก Data Register (D1) ส่งออกมาให้ MIC-SMART ผ่าน CH2 : RS232

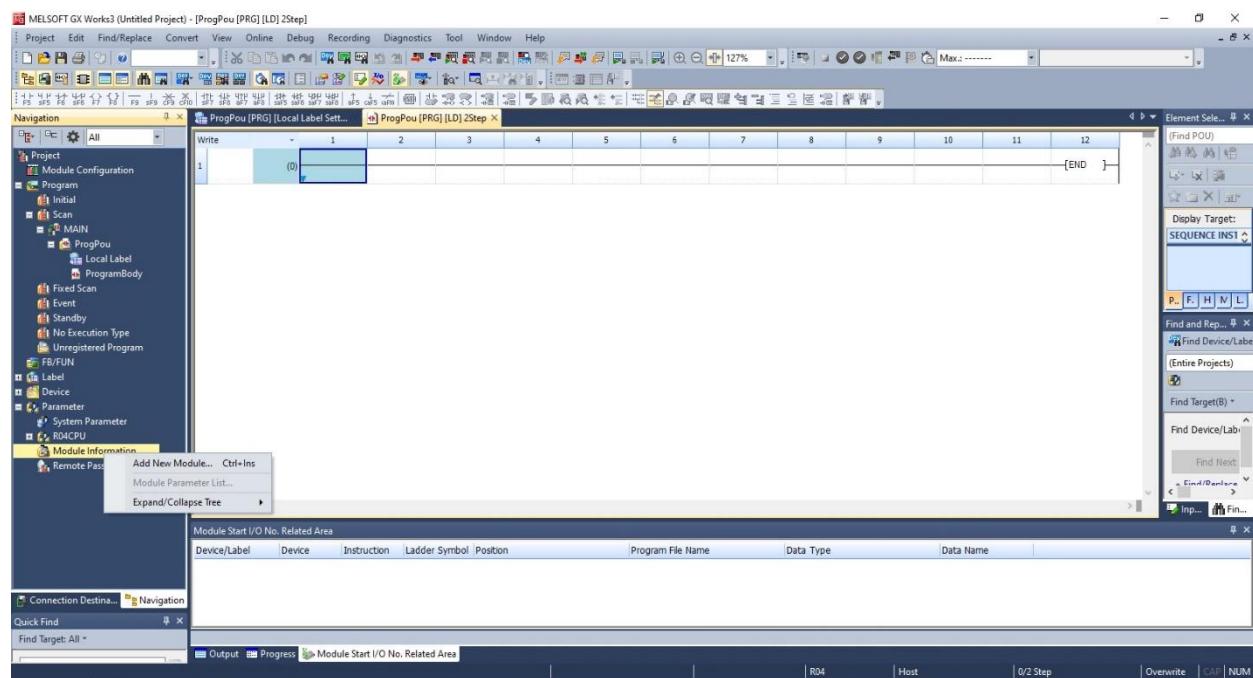
## 2.4 Serial Modbus RTU PLC

ในรูปเป็น PLC Mitsubishi CPU:R04 มีโมดูลสื่อสาร RJ71C24 ใช้สำหรับลื่อสารกับอุปกรณ์ MIC-SMART ใน CH1:RS232 ต่อใช้งานได้ดังรูป ( RS232 Cable ต้องใช้ชนิดที่เป็นตัวผู้ )



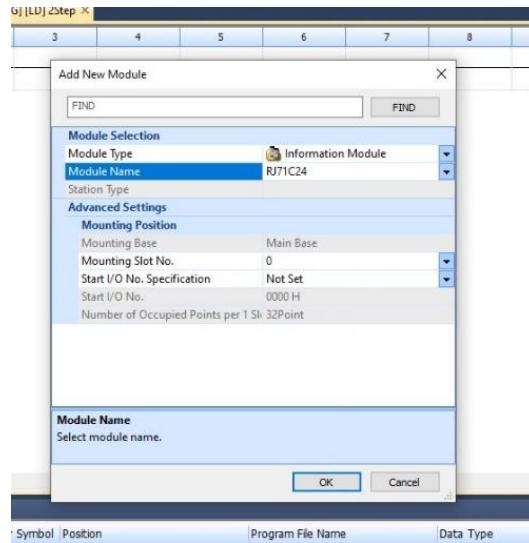
## ตัวอย่างวิธีเซตพารามิเตอร์ใน GX-WORK3

### เปิดสร้าง Project เรียบร้อยแล้ว ให้ทำการ Add New Module

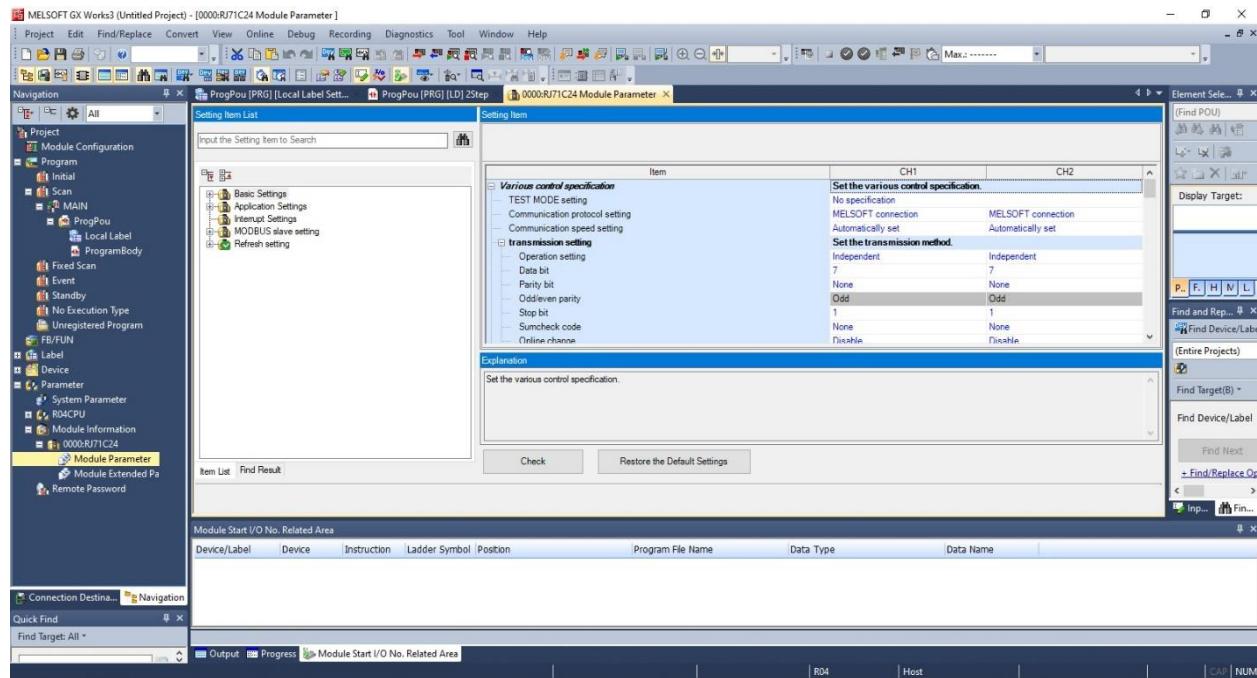


เลือก Module Type เป็น Information Module และ Module Name เป็น RJ71C24 จากนั้นกด

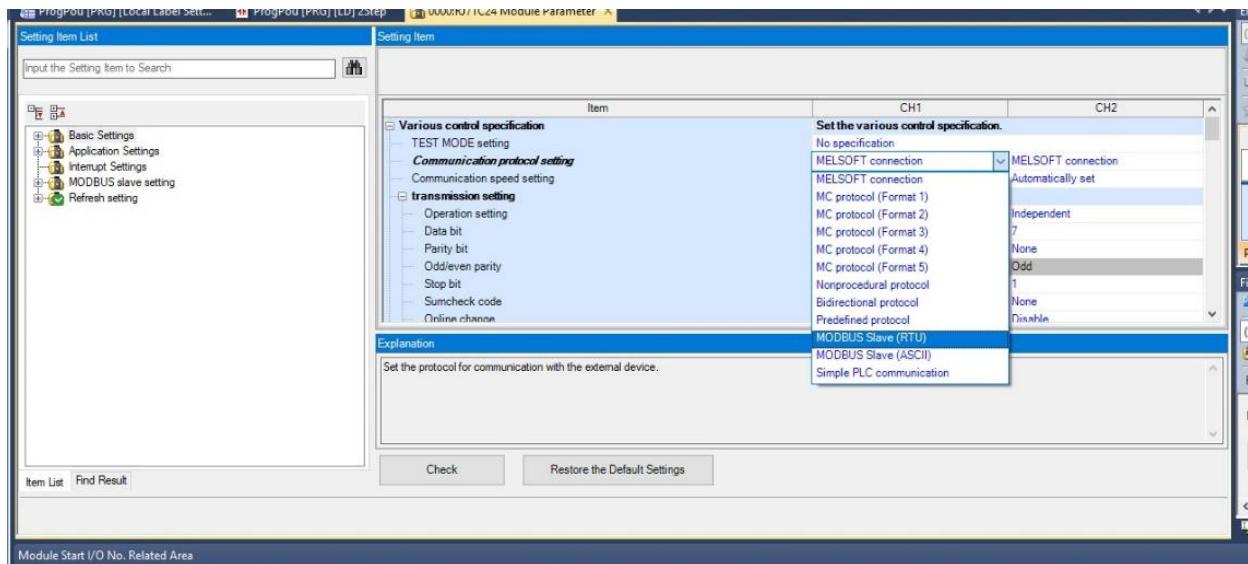
OK



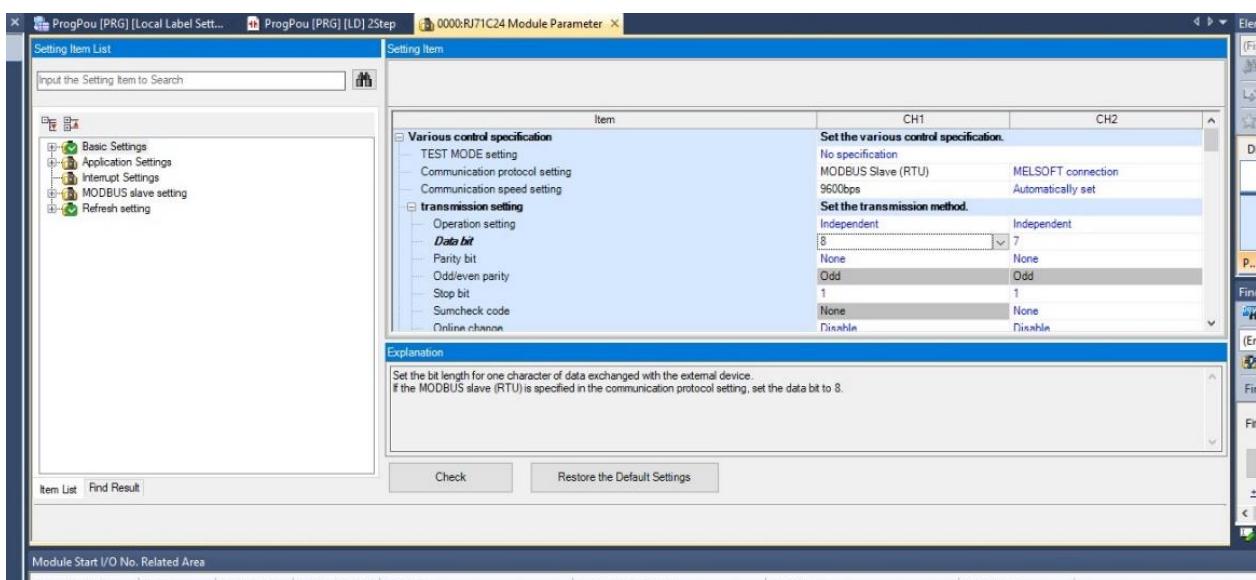
จากนั้นเลือก Module Parameter แล้วไปที่ Basic Setting



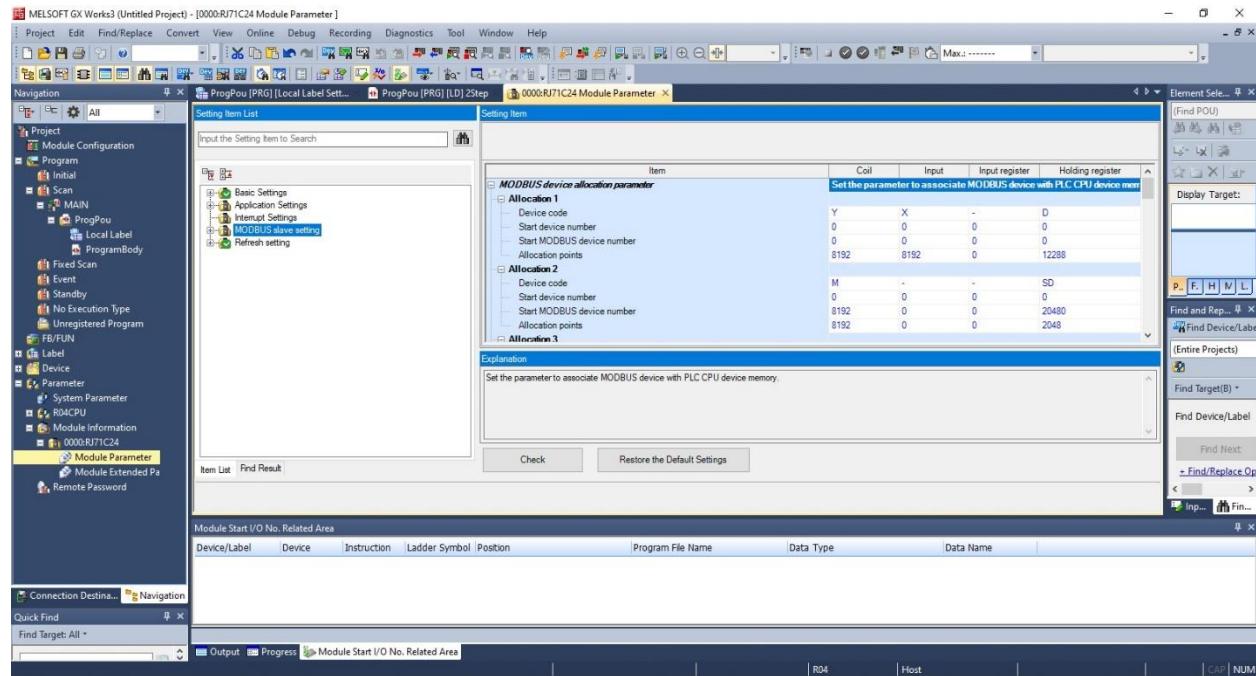
เปลี่ยน Communication protocol setting เป็น MODBUS Slave(RTU)



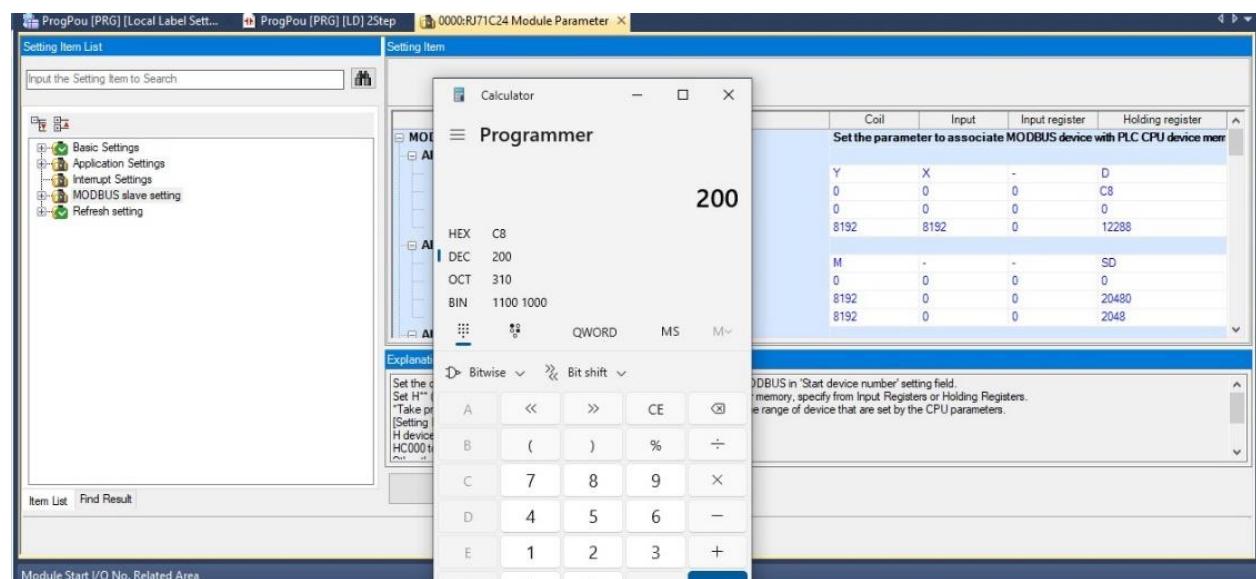
เลือก Communication speed setting เป็น 9600bps ( เช็ตตามกำหนดไว้ที่ MIC-SMART)  
และเปลี่ยน Data bit เท่ากับ 8



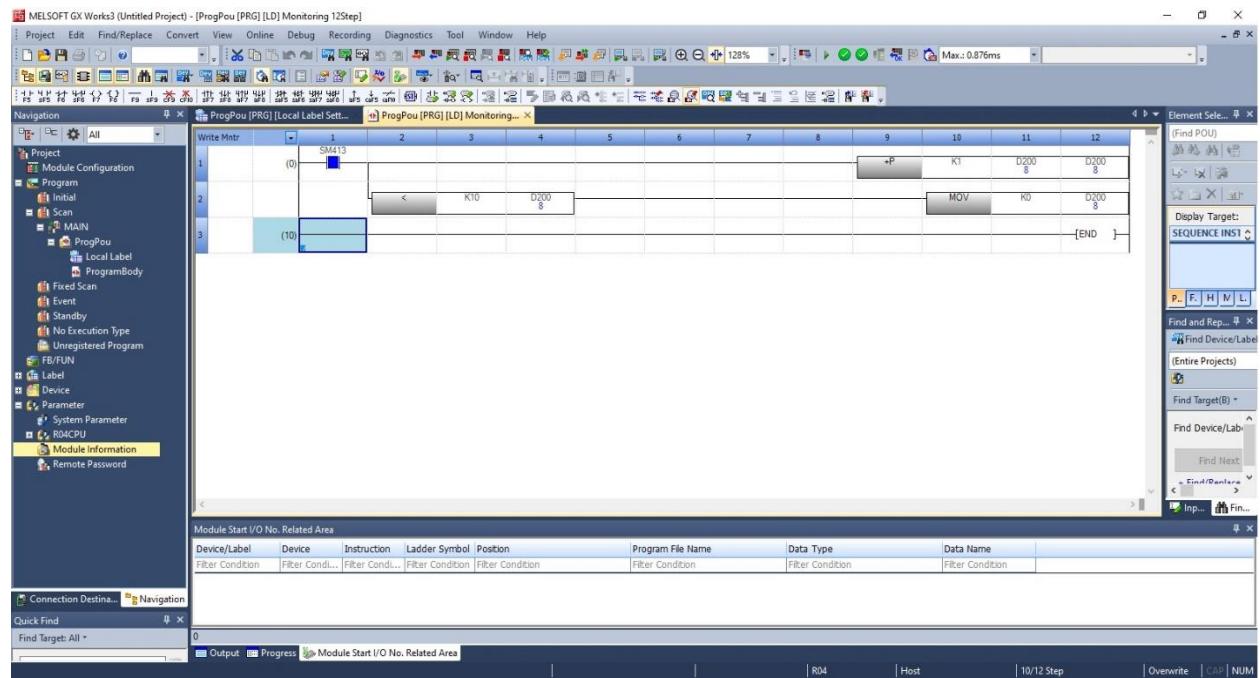
จากนั้นไปที่ MODBUS slave setting และเลือกที่ Start device number ในช่องของ Holding register เพื่อทำการเปลี่ยน Data Register เริ่มต้น



ในที่นี้กำหนด Data register เริ่มต้นที่ D200 ในช่องที่ใส่ค่าต้องเปลี่ยน 200 เลขฐาน10 เป็นเลขฐาน16 ได้ C8 นำค่า C8 ใส่ลงใน Start device number ในช่องของ Holding register



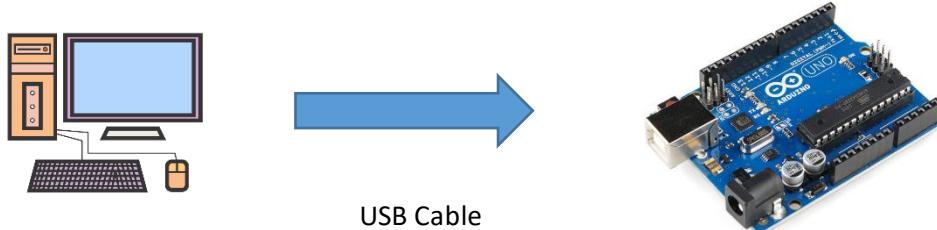
ตัวอย่าง Ladder ในการจำลองนับค่าที่ละ 1 ไปเก็บไว้ใน D200 และเมื่อ D200 มากกว่า 10 ให้ D200 เท่ากับ 0 ทำงานซ้ำไปเรื่อยๆ เพื่อทำการส่งค่าผ่าน Modbus



## CHAPTER 3: FEATURES OF Arduino IDE and Installation

### 3.1 FEATURES OF Arduino IDE

โปรแกรม Arduino IDE เป็นโปรแกรมที่ใช้สำหรับเขียนและอัปโหลดโค้ดลงบอร์ด Arduino โดยมีลักษณะเป็นโปรแกรมโต้ตอบ (Interactive program) ที่ช่วยให้ผู้ใช้งานสามารถเขียนโค้ดได้ง่ายและรวดเร็ว โดยมีคุณสมบัติหลากหลายรวมทั้งการอัปโหลดโค้ดผ่านทางพอร์ต USB และการตรวจสอบข้อผิดพลาด (Debugging) โดยโปรแกรมนี้รองรับการเขียนโค้ดโดยใช้ภาษา C++ และเพื่อความสะดวกสำหรับผู้ใช้งาน มีหลายตัวช่วย (Library) ที่เขียนไว้เพื่อช่วยในการเขียนโค้ดสำหรับงานต่าง ๆ



### ตัวอย่าง โปรแกรม Arduino IDE

```

sketch_mar11a | Arduino 1.8.19
File Edit Sketch Tools Help
void setup() {
1  // put your setup code here, to run once:
2
3 }
4
5
6void loop() {
7  // put your main code here, to run repeatedly:
8
9 }

```

The screenshot shows the Arduino IDE interface with the title bar "sketch\_mar11a | Arduino 1.8.19". The main window displays the following C++ code:

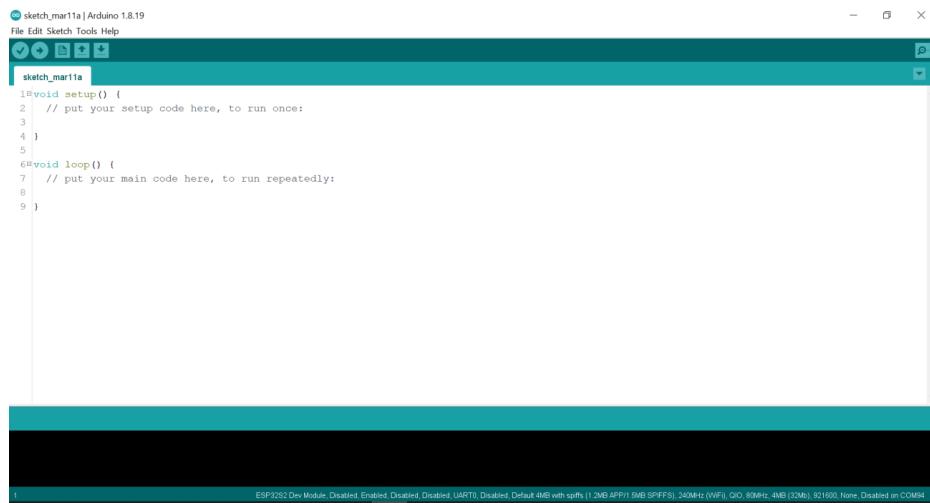
```

void setup() {
1  // put your setup code here, to run once:
2
3 }
4
5
6void loop() {
7  // put your main code here, to run repeatedly:
8
9 }

```

The status bar at the bottom indicates: "ESP32S2 Dev Module: Disabled, Enabled, Disabled, Disabled, UART0: Disabled, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), 240MHz (WiFi), QIO, 80MHz, 4MB (32Mb), 921600, None, Disabled on COM94".

## Arduino IDE Structure



1. Verify button.

Verify Checks your code for errors compiling it.

2. Upload button.

Upload Compiles your code and uploads it to the configured board.

3. New file.

New Creates a new sketch.

4. Open file.

Open presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window overwriting its content.

5. Save file.

Save your sketch.

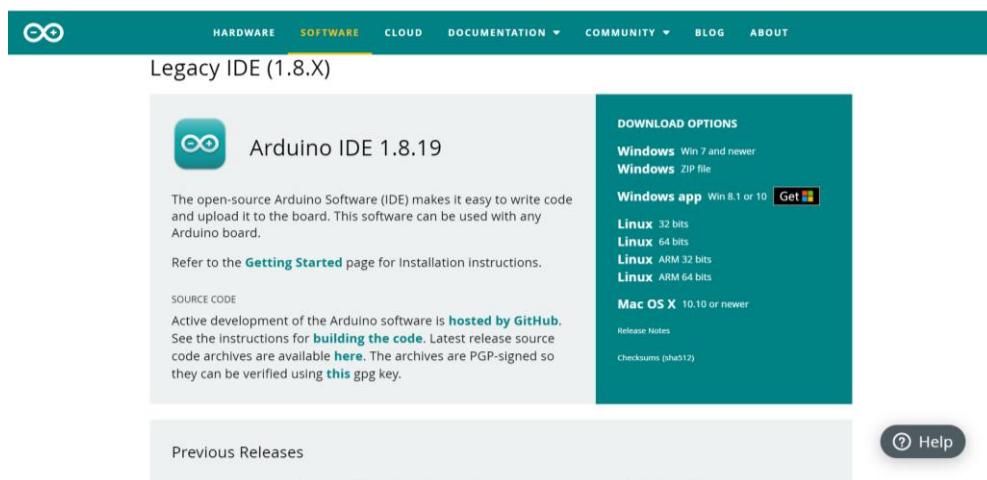
6. Serial Monitor button.

Serial Monitor Opens the serial monitor.

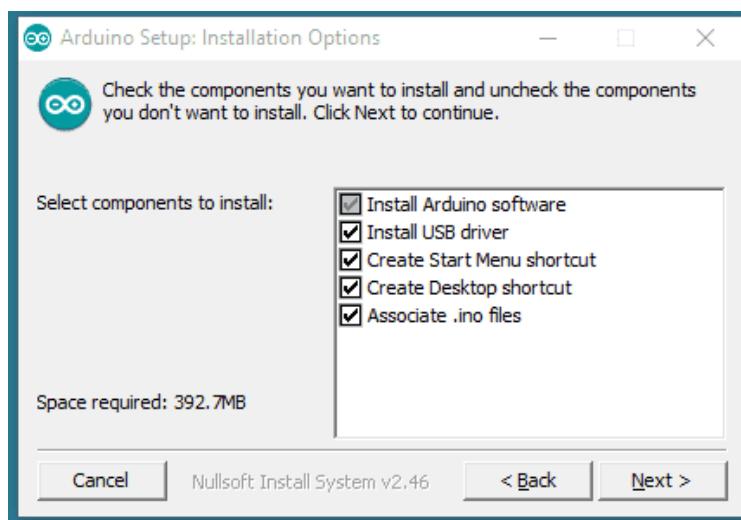
### 3.2 Installation Arduino IDE

Download Software Arduino IDE Version 1.8.19

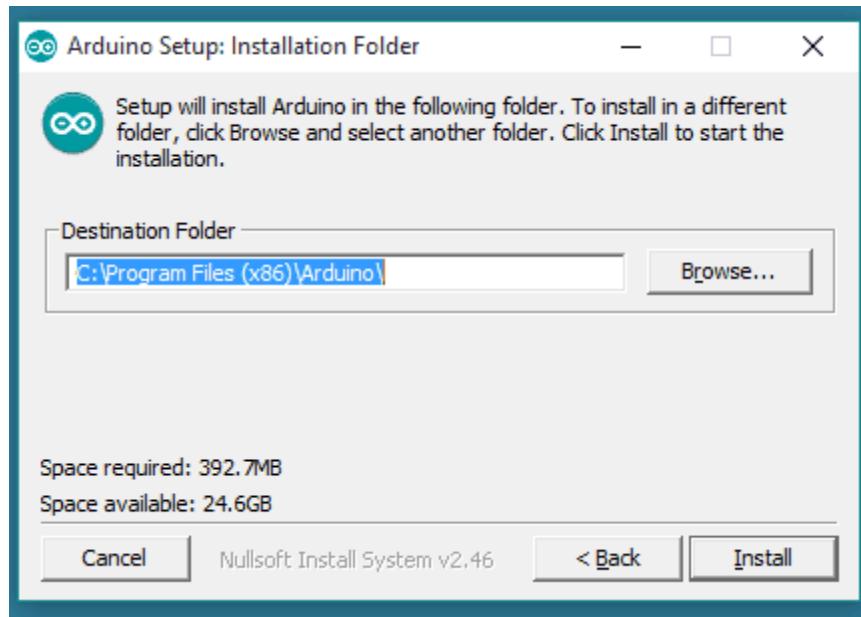
Get the latest version from the download Web <https://www.arduino.cc/en/software>. You can choose between the Installer (.exe) and the Zip packages.



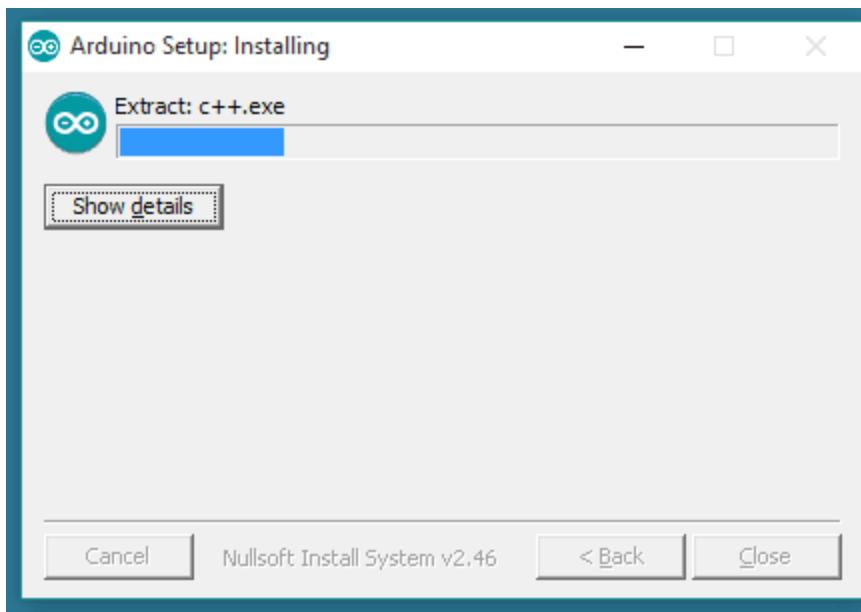
When the download finishes, proceed with the installation and please allow the driver installation process when you get a warning from the operating system.



Choose the components to install.



Choose the installation directory.

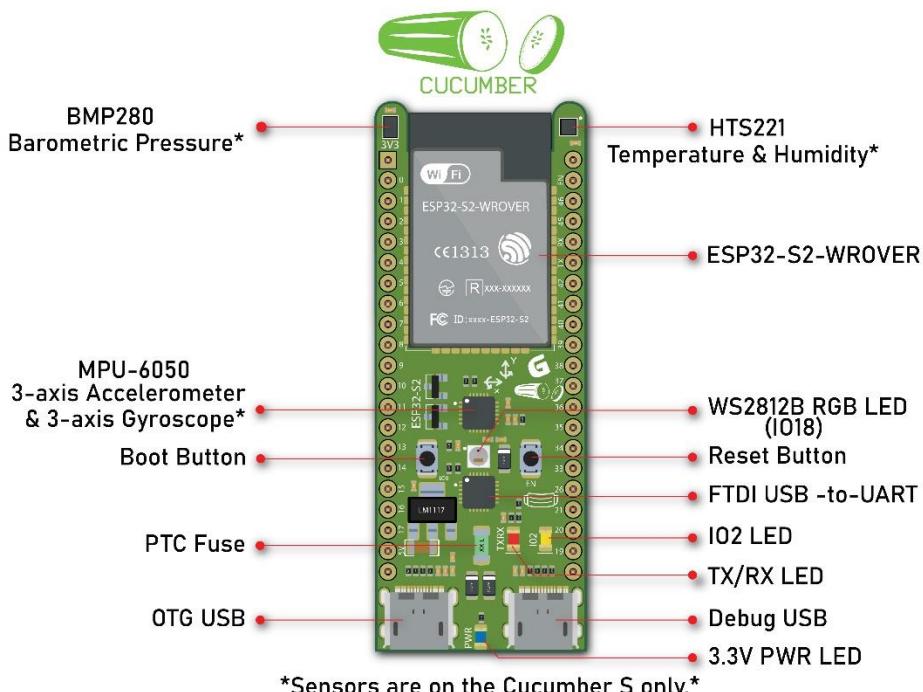


Installation in progress.

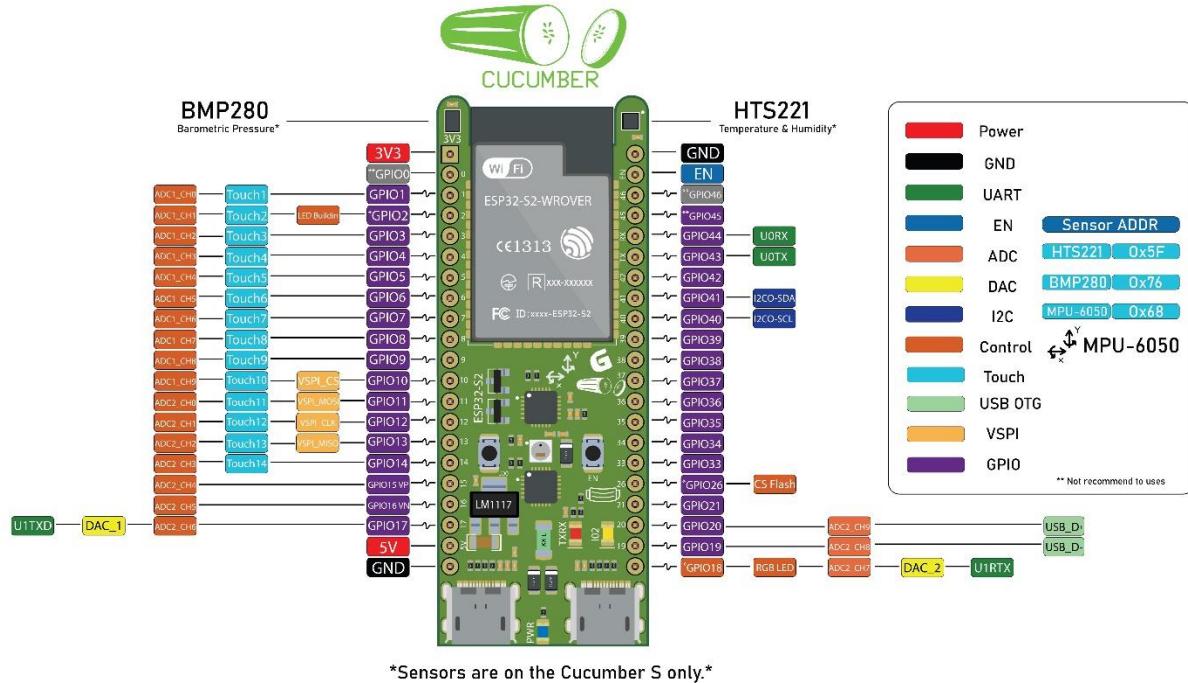
The process will extract and install all the required files to execute properly the Arduino Software (IDE)

### 3.3 ESP32S2 CUCUMBER Hardware

บอร์ด Cucumber RIS จะเป็นบอร์ดที่ใช้โมดูล ESP32-S2-WROVER ซึ่งมีความสามารถต่างๆ มากมาย เช่นการเรียนรู้ในการสร้างอุปกรณ์ IoT ได้ในราคาย่อมเยา ความสามารถของบอร์ดมีอาทิเช่น USB Type-C สองชุด เป็น Debug และ OTG พอร์ต, ใช้ FTDI ชิปเป็นตัว USB-to-Serial ซึ่งมีความสามารถและรองรับคอมพิวเตอร์ทุกรุ่น, RGB LED แบบใช้ขาสัญญาณเดินเดียว ผสมสีได้มากกว่า 16ล้านสี, มี TX/RX, IO2 และ PWR LED อีกทั้งมีเซนเซอร์ Barometric Pressure(BMP280), Temperature & Humidity(HTS221) และ 3-axis Accelerometer&3-axis Gyroscope(MPU-6050) ให้ใช้งานด้วย



## Cucumber Hardware Overview GRAVITECH



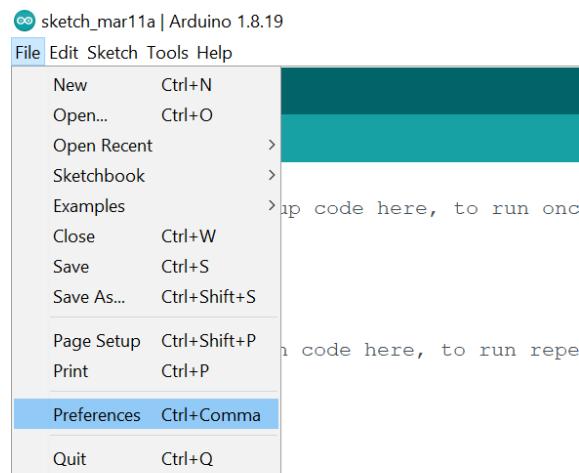
## CUCUMBER PINOUT

GRAVITECH

### Cucumber Pinout

## Installing ESP32S2 DEV Boards.

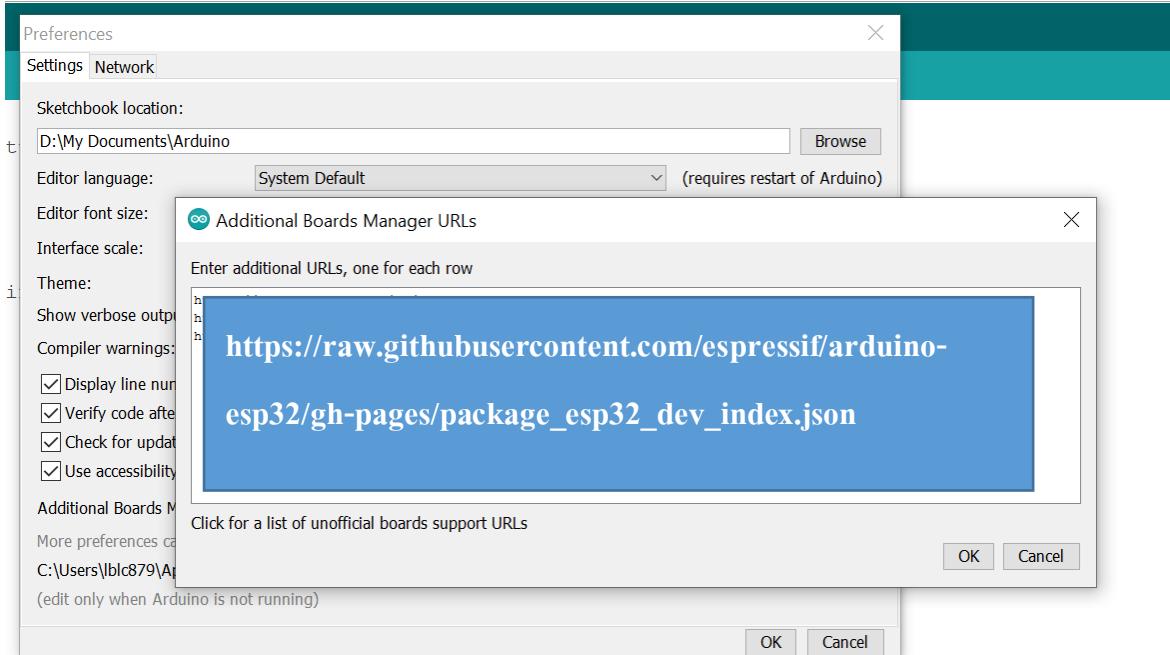
- Select File ----> Preference



- Update Additional Board

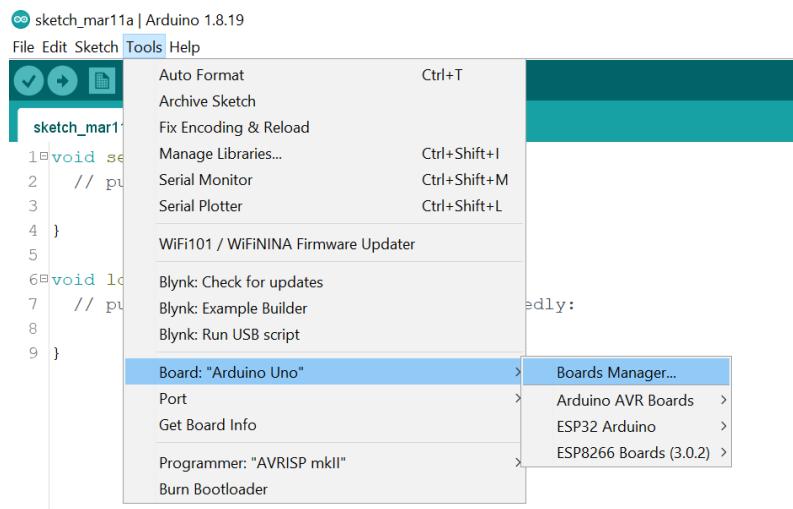
- OK

Add URL [https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package\\_esp32\\_dev\\_index.json](https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_dev_index.json)



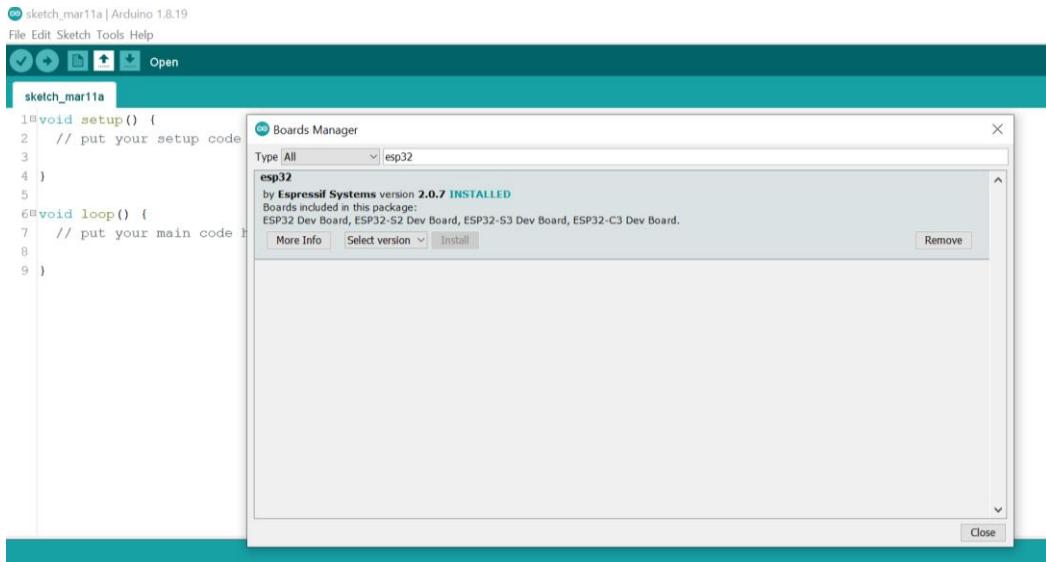
### Additional Board Manager URLs

- Tools ---> Board ---> Board Manager.

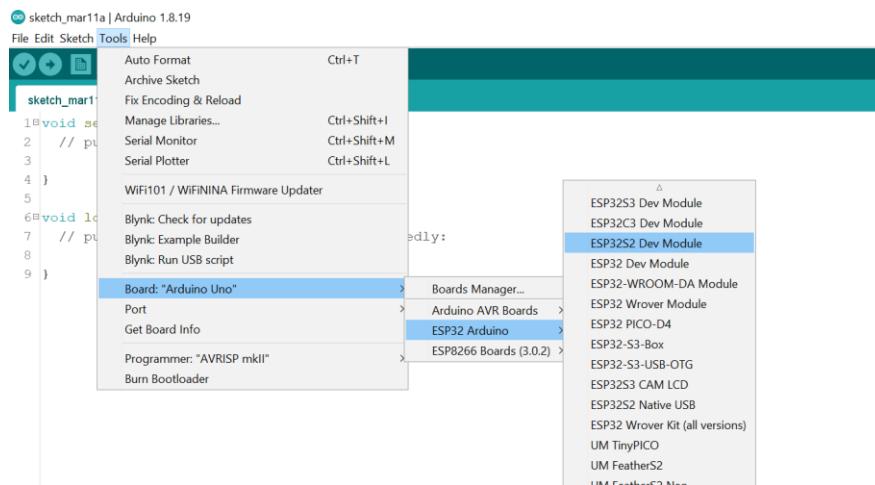


### Boards Manager

This will open up a new window, with all available cores. Find the one named **ESP32** and install it.

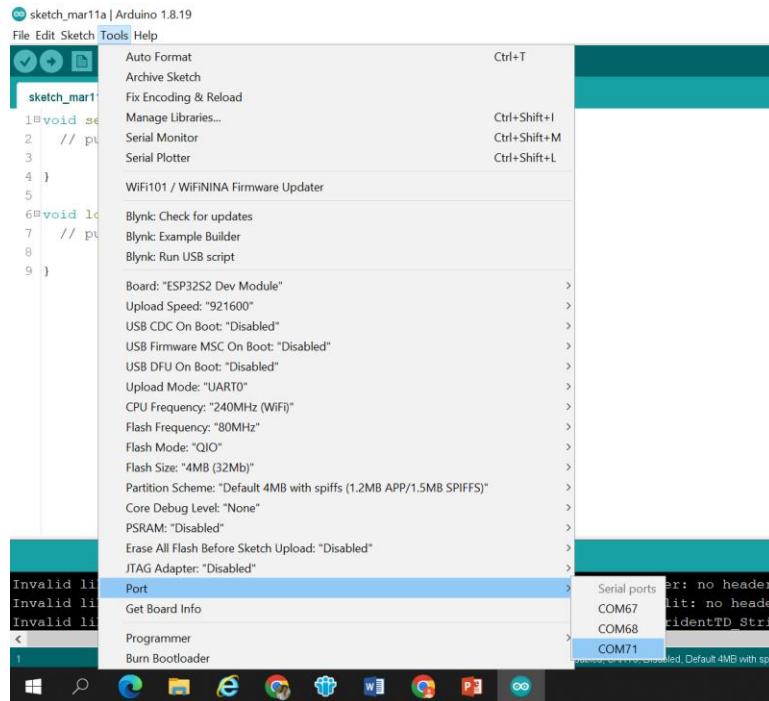


Exit the board manager, and go to **Tools > Board > ESP32 Arduino > ESP32S2 Dev Module** Here you can see all the ESP32S2 boards listed, where you can select the board you are using. You have now successfully installed the core.



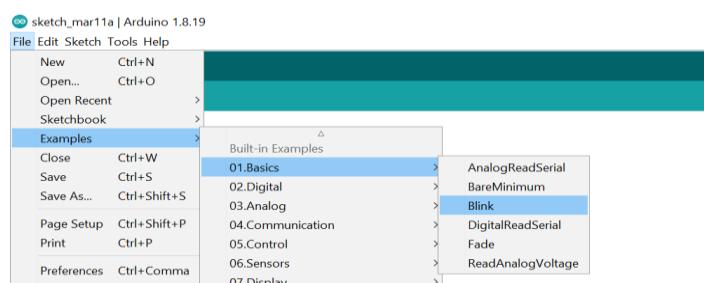
List of available boards.

Now, let's make sure that our board is found by our computer, by selecting the port. Regardless what kind of program we are uploading to the board, we **always** need to choose the port for the board we are using. This is simply done by navigating to **Tools > Port**, where you select your board from the list.



Selecting the right board and port.

You are now ready to start using your board! The easiest way to check that everything is working, is to upload just a simple blink example to your board. This is done by navigating to **File > Examples > 01.Basics > Blink.**



Selecting the blink example.

To upload the sketch, simply click on the arrow in the top left corner. This process takes a few seconds, and it is important to not disconnect the board during this process.



Uploading the sketch.

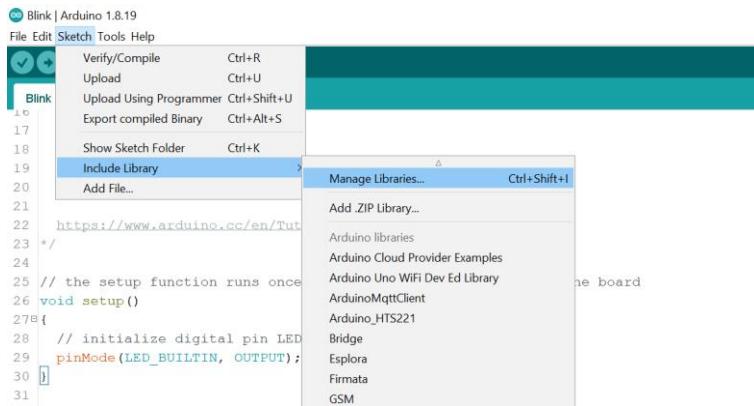
When the code is uploaded, the text "**Done uploading.**" is visible in the bottom left corner.

If you look closely at your board, you will notice an RGB LED blink with an interval of one second. This means you have successfully uploaded a program to your board.

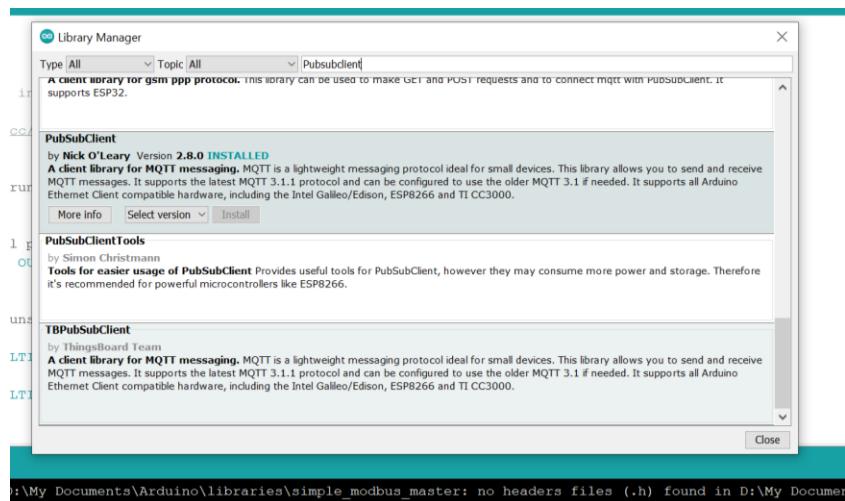
### 3.4 Installation Libraries

#### Using the Library Manager

To install a new library into your Arduino IDE you can use the Library Manager (available from IDE version 1.6.2). Open the IDE and click to the "Sketch" menu and then ***Include Library > Manage Libraries.***



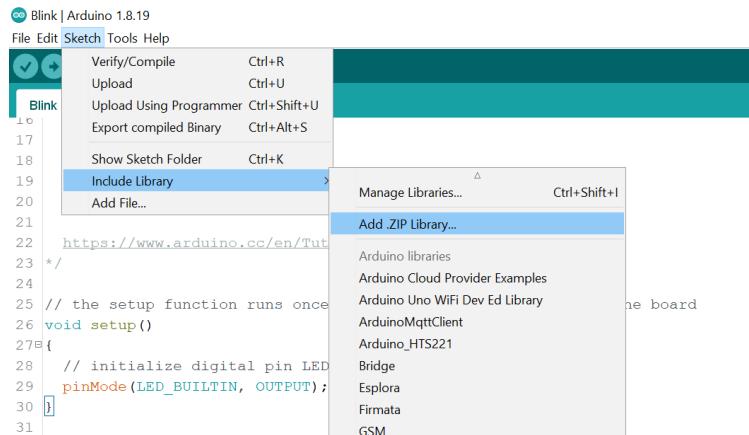
Then the Library Manager will open and you will find a list of libraries that are already installed or ready for installation. In this example we will install the PubSubClient library. Scroll the list to find it, click on it, then select the version of the library you want to install.



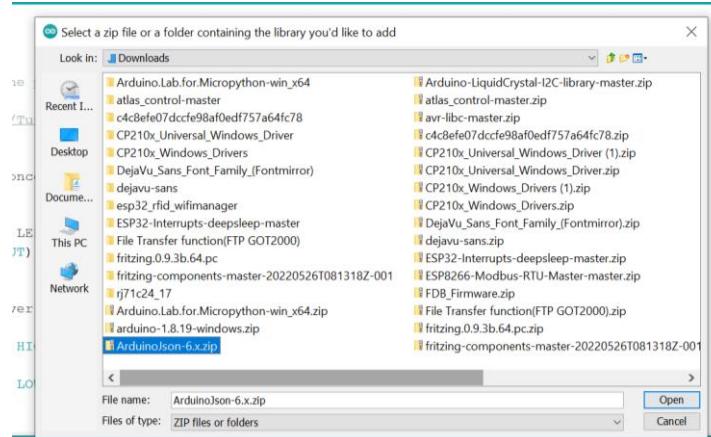
Finally click on install and wait for the IDE to install the new library. Downloading may take time depending on your connection speed. Once it has finished, an *Installed* tag should appear next to the PubSubClient library. You can close the library manager.

## Importing a .zip Library

In the Arduino IDE, navigate to *Sketch > Include Library > Add .ZIP Library*. At the top of the drop down list, select the option to "Add .ZIP Library".



You will be prompted to select the library you would like to add. Navigate to the .zip file's location and open it.

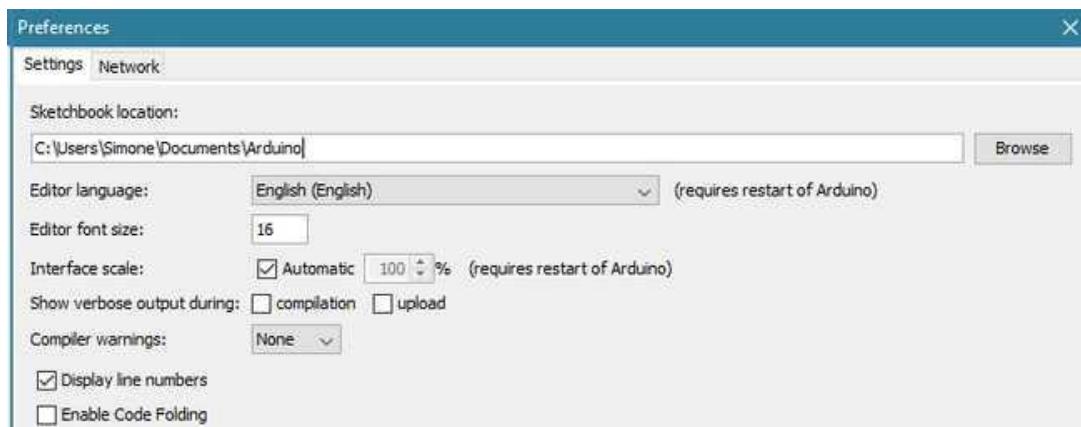


Return to the *Sketch > Include Library* menu. You should now see the library at the bottom of the drop-down menu. It is ready to be used in your sketch. The zip file will have been expanded in the *libraries* folder in your Arduino sketches directory.

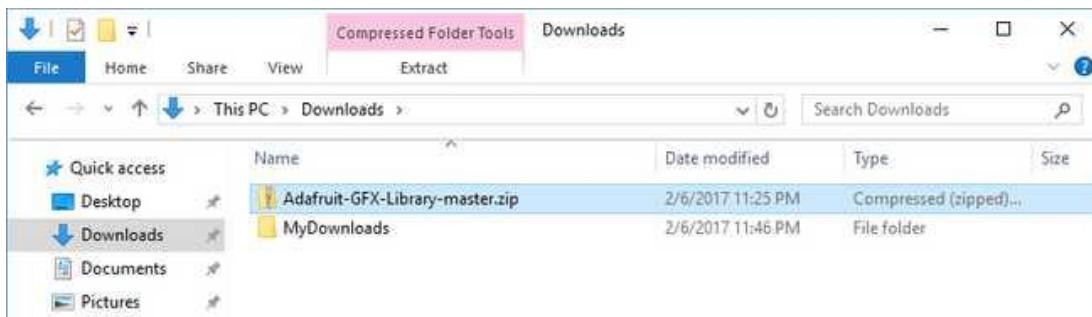
## Manual Installation

When you want to add a library manually, you need to download it as a ZIP file, expand it and put in the proper directory.

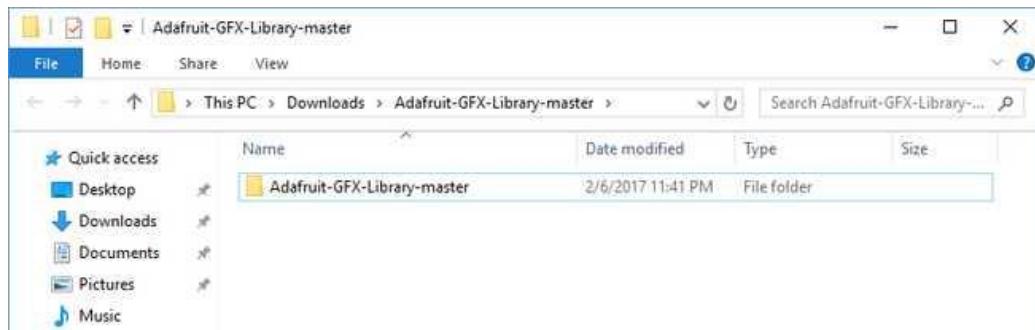
You can find or change the location of your sketchbook folder at *File > Preferences > Sketchbook location*.



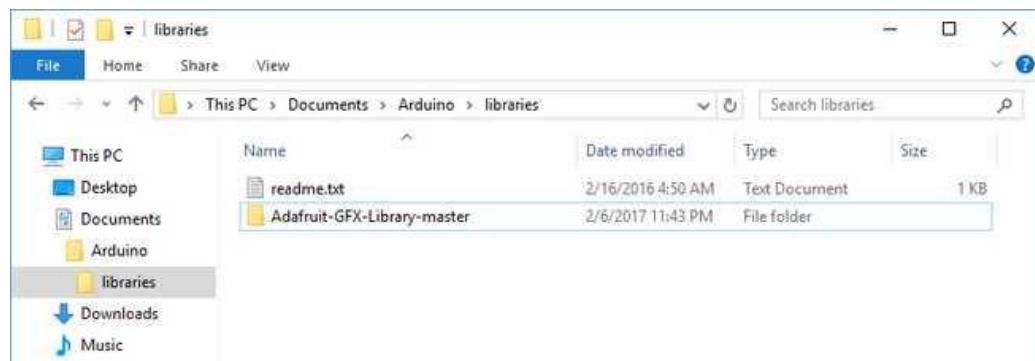
Go to the directory where you have downloaded the ZIP file of the library



Extract the ZIP file with all its folder structure in a temporary folder, then select the main folder, that should have the library name



Copy it in the "libraries" folder inside your sketchbook.

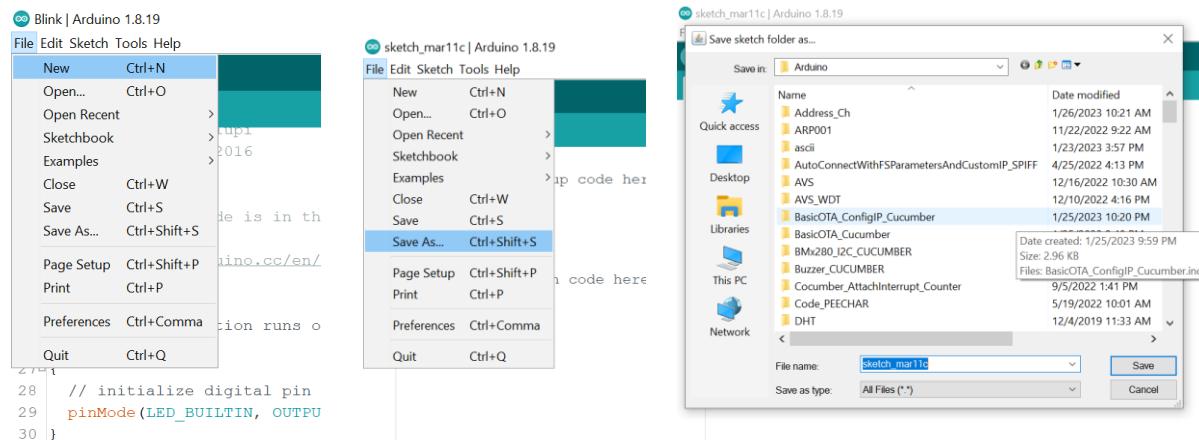


Start the Arduino Software (IDE), go to *Sketch > Include Library*. Verify that the library you just added is available in the list.

### 3.5 Create New Project

Create New Project File--->New

Save Project File--->Save As..



### 3.6 C++ language structure used in the Arduino IDE

The Arduino IDE is a tool used for writing programs for the Arduino board using the C++ language. Writing programs in the Arduino IDE follows a similar structure to writing programs in C++, but there are additional libraries and special functions specifically for programming the Arduino board.

The key elements of the C++ language structure used in the Arduino IDE include:

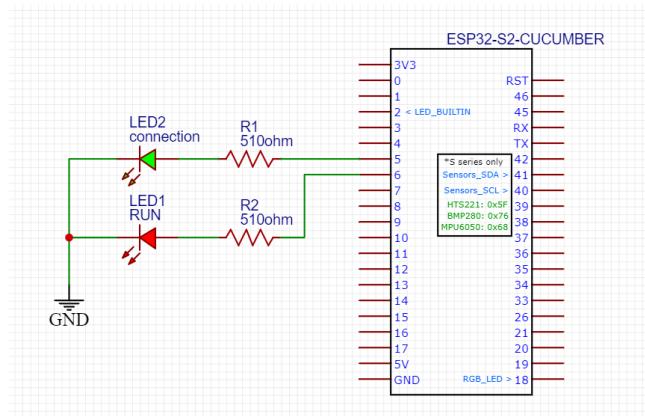
- Libraries - The Arduino IDE has a wide range of libraries developed to make programming the Arduino board easier, such as Wire.h for using I2C, Servo.h for controlling servo motors, and many more.
- Main function - The main function in the Arduino IDE is similar to C++, but it calls the setup() and loop() functions to set up and control the operation of the Arduino board. Here is an example of an Arduino IDE program:

```
sketch_mar16a §
1 #include <Servo.h>
2
3 Servo myservo; // create servo object t
4
5 void setup() {
6   myservo.attach(9); // attaches the se
7 }
8
9 void loop() {
10   myservo.write(90); // sets the servo
11   delay(1000); // waits for a sec
12   myservo.write(0); // sets the servo
13   delay(1000); // waits for a sec
14 }
```

## CHAPTER 4: INTRODUCTION TO MIC SMART

### 4.1 การตั้งค่า LED

ในตัวอย่างนี้จะทดลองตั้งค่า LED Connection กับ Run ซึ่งได้ออกแบบให้ led ของ Connection ต่ออยู่กับ GPIO5 ส่วน led ของ Run ต่ออยู่กับ GPIO6



แสดงการต่อวงจร LED Published data กับ Connection

### ตัวอย่างทดลองเปลี่ยนโปรแกรมตั้งค่า LED

LCD

```

1 #define led_connection 5
2 #define led_run 6
3
4 void setup() {
5   pinMode(led_connection, OUTPUT);
6   pinMode(led_run, OUTPUT);
7 }
8
9
10 void loop() {
11 {
12   digitalWrite(led_connection, HIGH);
13   digitalWrite(led_run, HIGH);
14   delay(1000);
15   digitalWrite(led_connection, LOW);
16   digitalWrite(led_run, LOW);
17   delay(1000);
18 }

```

```
#define led_connection 5 กำหนดขาสั่งงาน
#define led_run 6 กำหนดขาสั่งงาน
pinMode(led_connection, OUTPUT);
กำหนดให้ led_connection เป็น OUTPUT
pinMode(led_run, OUTPUT);กำหนดให้ led_
published เป็น OUTPUT
digitalWrite(led_connection,HIGH); สำหรับ
led_connection ทำงาน หากเปลี่ยนจาก HIGH
เป็น LOW จะหยุดทำงาน

```

## 4.2 การสร้าง Function.

การสร้างฟังก์ชัน (Function) เป็นการกำหนดชุดของคำสั่งที่จะทำงานเมื่อถูกเรียกใช้ โดยใช้รูปแบบการสร้างฟังก์ชันดังนี้

### Function

```

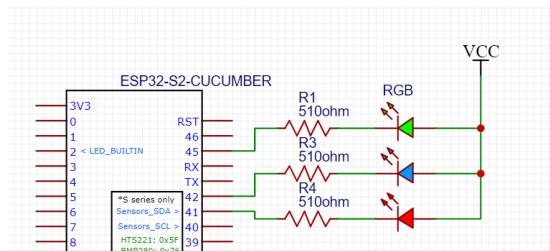
1 #define led_connection 5
2 #define led_run 6
3
4 void led_connect()
5 {
6     digitalWrite(led_connection, HIGH);
7     delay(1000);
8     digitalWrite(led_connection, LOW);
9     delay(1000);
10 }
11
12 void setup() {
13     pinMode(led_connection, OUTPUT);
14     pinMode(led_run, OUTPUT);
15 }
16
17 void loop()
18 {
19     led_connect();
20 }
21

```

ใน void loop() เรียกฟังก์ชัน led\_connect(); มาใช้งาน  
จึงทำให้ใน void led\_connect() ทำงาน สั่งการให้ led\_connection กระพริบทุกๆ 1 วินาที

## 4.3 RGB LED

Light Emitting Diode สีแดง เขียว และน้ำเงิน ซึ่งเป็นชนิดของ ไดโอดที่สามารถแสดงสีได้ หลากหลายโดยการเปลี่ยนความเข้มของแสงสีแดง เขียว และน้ำเงิน ซึ่งแต่ละสีมีความยาวคลื่นแสงที่แตกต่างกัน และเมื่อร่วมกัน ได้ผลลัพธ์เป็นสีต่างๆ ซึ่งกำหนดให้ สีแดง:GPIO41, สีเขียว:GPIO45, สีน้ำเงิน:GPIO42



แสดงการต่อวงจร RGB LED

## ตัวอย่างทดลองเปลี่ยน โปรแกรมสั่งงาน RGB LED

```

RGB
1 #define rgb_red 41
2 #define rgb_green 45
3 #define rgb_blue 42
4
5 void setup()
6{
7  pinMode(rgb_red, OUTPUT);
8  pinMode(rgb_green, OUTPUT);
9  pinMode(rgb_blue, OUTPUT);
10
11}
12
13 void loop()
14{
15  led_red();
16  delay(1000);
17  led_green();
18  delay(1000);
19  led_blue();
20  delay(1000);
21  led_off();
22  delay(1000);
23}
24
25
26 void led_red()
27{
28  digitalWrite(rgb_red, LOW); digitalWrite(rgb_green, HIGH); digitalWrite(rgb_blue, HIGH);
29}
30
31 void led_green()
32{
33  digitalWrite(rgb_red, HIGH); digitalWrite(rgb_green, LOW); digitalWrite(rgb_blue, HIGH);
34}
35
36 void led_blue()
37{
38  digitalWrite(rgb_red, HIGH); digitalWrite(rgb_green, HIGH); digitalWrite(rgb_blue, LOW);
39}
40
41 void led_off()
42{
43  digitalWrite(rgb_red, HIGH); digitalWrite(rgb_green, HIGH); digitalWrite(rgb_blue, HIGH);
44}

```

กำหนดให้ `rgb_red : GPIO41, rgb_green : GPIO45, rgb_blue:GPIO42` ทั้งหมดนี้กำหนดให้เป็น OUTPUT การทำงานใน `loop()`; จะทำการเรียกใช้ Function `led_red()`; เพื่อให้ RGB เป็นสีแดงและหน่วงเวลา 1000mS จากนั้นเรียกใช้ `led_green()`; `led_blue()`; และ `led_off()`; โดยแต่ละการทำงานของ Function จะหน่วงเวลา 1000mS

ผลที่ได้ RGB LED จะเปลี่ยนสีแดง, สีเขียว, สีฟ้าเงิน และดับทุกๆ 1 วินาที วนซ้ำไปเรื่อยๆ

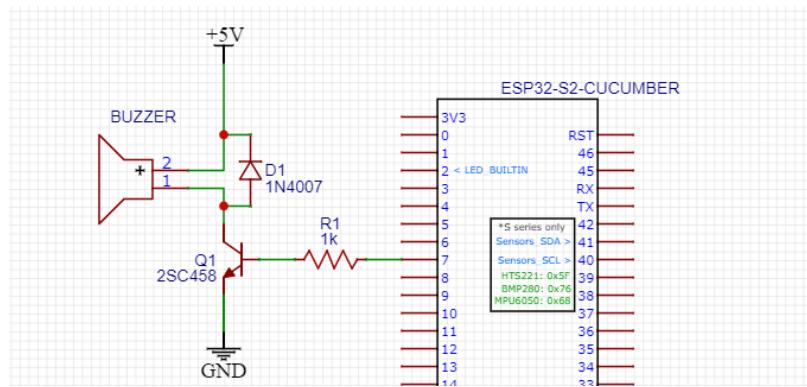
## 4.4 Buzzer

Buzzer เป็นอุปกรณ์ส่งเสียงที่มีลักษณะเป็นลักษณะเสียงที่ออกมากเป็นเสียงกระแทบ นิยมนำมาใช้ในการแจ้งเตือน เช่น เมื่อเกิดเหตุฉุกเฉิน หรือในการแจ้งเตือนการเข้าใช้งานอุปกรณ์ที่มีความเสี่ยงสูง เช่น เตือนเมื่อเปิดเครื่องจักรอัตโนมัติ นอกจากนี้ยังมีการนำ buzzer ไปใช้งานต่างๆ เช่น การต่อเสียงเป็นตัวบอกเวลา หรือเป็นส่วนประกอบในงานอิเล็กทรอนิกส์ต่างๆ ด้วย

## Active Buzzer 5V TMB12A05



รูปตัวอย่าง Buzzer 5V



แสดงการต่อวงจรสั่งงาน Buzzer

ตัวอย่างทดลองเขียนโปรแกรมสั่งงาน Buzzer

### Buzzer

```

1 #define buzzer 7
2
3 void setup()
4 {
5   pinMode(buzzer, OUTPUT);
6 }
7
8 void loop()
9 {
10  digitalWrite(buzzer, HIGH);
11  delay(100);
12  digitalWrite(buzzer, LOW);
13  delay[5000];
14 }
15

```

กำหนดให้ buzzer : GPIO7 ให้เป็น

OUTPUT การทำงานคือสั่งให้ buzzer

ทำงานโดยหน่วงเวลา 100mS และหยุดการ

ทำงาน 5000mS วนซ้ำไปเรื่อยๆ

## 4.5 Serial Monitor

Serial Monitor ใน Arduino IDE เป็นเครื่องมือที่ช่วยให้คุณสื่อสารกับบอร์ด Arduino ของคุณผ่านพอร์ตซีเรียลได้อย่างง่ายดาย โดยคุณสามารถส่งและรับข้อมูลระหว่างคอมพิวเตอร์และบอร์ด Arduino ในเวลาจริงได้ คุณสามารถใช้ Serial Monitor เพื่อทำการดังนี้

1. สามารถพิมพ์ข้อมูลเดิมกลับใน Serial Monitor เพื่อช่วยคุณเข้าใจว่าโค้ดทำงานอย่างไรและระบุข้อผิดพลาด
2. สามารถส่งคำสั่งและข้อมูลไปยังบอร์ด Arduino ผ่าน Serial Monitor เพื่อทดสอบโค้ดและตรวจสอบว่ามันทำงานตามที่ต้องการหรือไม่
3. ตรวจสอบข้อมูลเซ็นเซอร์ที่ต่อ กับบอร์ด Arduino สามารถใช้ Serial Monitor เพื่อดูข้อมูลเซ็นเซอร์ในเวลาจริงได้

เพื่อเปิด Serial Monitor ต้องอัปโหลดสคริปต์ลงบนบอร์ด Arduino รวมฟังก์ชัน Serial.begin() หลังจากนั้นคลิกที่ปุ่ม Serial Monitor ใน Arduino IDE (ไอคอนแฉ่ง่ายในมุมขวาบนของหน้าต่าง) หรือไปที่ เครื่องมือ > Serial Monitor ในแถบเมนู จะเห็นหน้าต่างใหม่ที่แสดง Serial Monitor

```
Serial_Monitor
1 void setup()
2 {
3     Serial.begin(112500);
4     Serial.begin(9600);
5 }
6
7 void loop()
8 {
9     Serial.print("MIC-SMART-Serial");
10    Serial.print("MIC-SMART-Serial1");
11    delay(1000);
12 }
13
```

เปิดการทำงานฟังก์ชัน Serial.begin(9600);  
แล้วใน void loop() ให้ใช้คำสั่ง Serial.print เพื่อแสดง  
ข้อมูล MIC-SMART-Serial ใน Serial monitor

ส่วน Serial1 จะส่งข้อมูลที่พอร์ต RS232  
ผ่านสาย RS232 cable กับ USB to RS232 cable และทำการเลือก Port ที่ต้องการ Monitor ให้ถูกต้อง

## 4.6 ชนิดของข้อมูลและการใช้ตัวแปร

ชนิดข้อมูลหรือประเภทของข้อมูล เป็นตัวบอกรายละเอียดของตัวแปรที่ประกาศใช้งานว่า ตัวแปรดังกล่าวสามารถใช้เก็บข้อมูลอะไรได้ ในภาษาซี จะมีชนิดข้อมูลที่ประกอบไปด้วยตัวอักษร (Character) จำนวนตัวเลขแบบจานวนเต็ม (Integer) และจำนวนตัวเลขทศนิยม (Floating point number) ชนิดข้อมูลทั้ง 3 แบบสามารถแบ่งออกเป็นชนิดข้อมูลต่างๆ ได้ตามตาราง

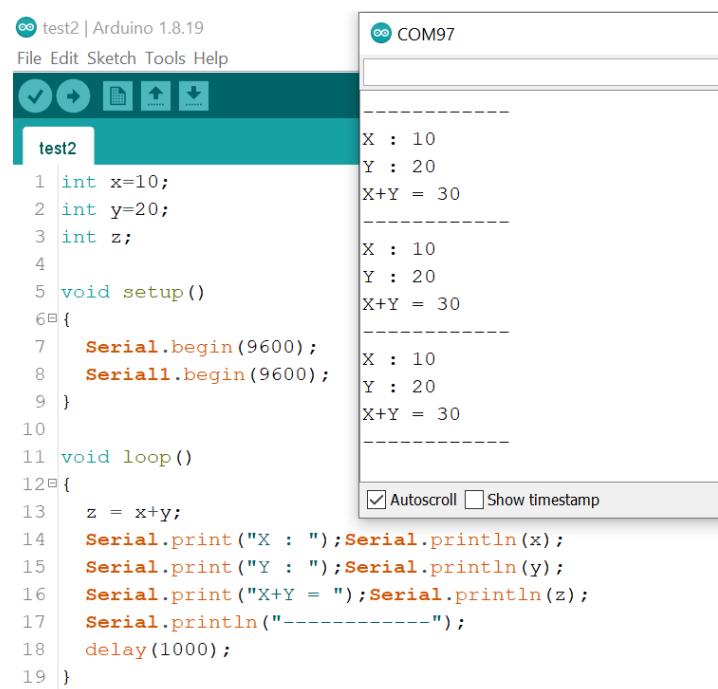
ชนิดข้อมูล	การเก็บข้อมูล	ขนาด
boolean	จริง (True) หรือ เท็จ (False)	1 บิต
char	ตัวเลข หรือตัวอักษร	1 "บิต" ใส่ค่าได้ตั้งแต่ -128 ถึง 127
unsigned char	ตัวเลข หรือตัวอักษร	1 "บิต" ใส่ค่าได้ตั้งแต่ 0 ถึง 255
byte	"บิต"	1 "บิต" ใส่ค่าได้ตั้งแต่ 0 ถึง 255
int	ตัวเลขจำนวนเต็ม	2 "บิต" ใส่ค่าได้ตั้งแต่ -32,768 ถึง 32,767
unsigned int	ตัวเลขจำนวนเต็ม	2 "บิต" ใส่ค่าได้ตั้งแต่ 0 ถึง 65,535 ( $2^{16}$ ) - 1
long	ตัวเลขจำนวนเต็มที่มีความยาว	4 "บิต" ใส่ค่าได้ตั้งแต่ -2,147,483,648 ถึง 2,147,483,647
unsigned long	ตัวเลขจำนวนเต็มที่มีความยาว	4 "บิต" ใส่ค่าได้ตั้งแต่ 0 ถึง 4,294,967,295 ( $2^{32}$ - 1)
float	ตัวเลขทศนิยมใช้ในการคำนวณ	4 "บิต" ใส่ค่าได้ตั้งแต่ 3.4028235E+38 ถึง -3.4028235E+38 มีทศนิยมได้ 6 ถึง 7 ตำแหน่ง
double (เฉพาะบอร์ด Arduino Due)	ตัวเลขทศนิยมที่มีความยาวและต้องการความแม่นยำ	8 "บิต" ใช้ในการคำนวณที่ต้องการประสิทธิภาพสูง
String	ข้อความ	ไม่ระบุ

### ตัวอย่างการนำประเภทของตัวแปรมาใช้งาน

1. กำหนดให้ int x=10;

int y=20;

int z=x+y;



```
test2 | Arduino 1.8.19
File Edit Sketch Tools Help
COM97
-----
X : 10
Y : 20
X+Y = 30
-----
X : 10
Y : 20
X+Y = 30
-----
X : 10
Y : 20
X+Y = 30
-----
Autoscroll Show timestamp

```

```

1 int x=10;
2 int y=20;
3 int z;
4
5 void setup()
6 {
7   Serial.begin(9600);
8   Serial1.begin(9600);
9 }
10
11 void loop()
12 {
13   z = x+y;
14   Serial.print("X : ");Serial.println(x);
15   Serial.print("Y : ");Serial.println(y);
16   Serial.print("X+Y = ");Serial.println(z);
17   Serial.println("-----");
18   delay(1000);
19 }
```

2. กำหนดให้ float x = 10;

float y = 20;

float z = x/y;

```

    test2 | Arduino 1.8.19
File Edit Sketch Tools Help
[checkmark] [refresh] [file] [upload] [down]
test2
1 float x=10,y=20,z;
2
3 void setup()
4 {
5     Serial.begin(9600);
6     Serial1.begin(9600);
7 }
8
9 void loop()
10 {
11     z = x/y;
12     Serial.print("X : ");Serial.println(x);
13     Serial.print("Y : ");Serial.println(y);
14     Serial.print("X/Y = ");Serial.println(z);
15     Serial.println("-----");
16     delay(1000);
17 }

```

COM97

```

X : 10.00
Y : 20.00
X/Y = 0.50
-----
X : 10.00
Y : 20.00
X/Y = 0.50
-----
X : 10.00
Y : 20.00
X/Y = 0.50
-----
X : 10.0

```

Autoscroll  Show timestamp

#### 4.7 ตัวแปรอาร์เรย์ (Array)

ตัวแปรอาร์เรย์ (array) ใน Arduino คือตัวแปรที่เก็บข้อมูลหลายค่าในตัวเดียว โดยสามารถเข้าถึงข้อมูลในอาร์เรย์ได้โดยการใช้ตัวชี้ (index) เป็นตัวเลขซึ่งเริ่มจาก 0 สูงสุดถึง n-1 (n คือจำนวนสมาชิกในอาร์เรย์) การประกาศตัวแปรอาร์เรย์ใน Arduino จะมีรูปแบบดังนี้

type name[size];

โดย type คือชนิดของข้อมูลในอาร์เรย์ เช่น int, float, char, และอื่น ๆ และ size คือขนาดของอาร์เรย์หรือจำนวนสมาชิกในอาร์เรย์ ที่ต้องการประกาศ

ตัวอย่างการประกาศตัวแปรอาร์เรย์ int ที่มีชื่อ myArray และมีขนาด 5

int myArray[5];

การเข้าถึงข้อมูลในอาร์เรย์จะใช้ตัวชี้ (index) โดยใช้วงเดือนเหลี่ยม [] โดยตัวชี้จะเริ่มจาก 0 สูงสุดถึง n-1 โดย n คือจำนวนสมาชิกในอาร์เรย์

ตัวอย่างการเข้าถึงข้อมูลในอาร์เรย์ myArray ที่ตัวชี้เป็น 2 ค่าของตัวแปร value จะเป็นค่าในตำแหน่งที่ 3 ของ myArray.

```
int value = myArray[2];
```

ตัวอย่างโปรแกรมประกาศตัวแปร Array

The screenshot shows the Arduino IDE interface. On the left, there is a code editor window titled "Array" containing the following C-like pseudocode:

```

1 int myArray[6] = {1, 3, 5, 7, 8, 9};
2
3 void setup()
4 {
5   Serial.begin(9600);
6   Serial1.begin(9600);
7 }
8
9 void loop()
10 {
11   Serial.print("myArray[2] : "); Serial.println(myArray[2]);
12   delay(1000);
13 }
14

```

On the right, there is a "Serial Monitor" window titled "COM103" showing the output of the code. It displays the value "myArray[2] : 5" repeated 12 times, corresponding to the 12 iterations of the loop.

นอกเหนือไปนี้ยังสามารถใช้งานอาร์เรย์เพื่อเก็บค่าตัวอักษร (String) หรือตัวแปรประเภทอื่นๆ ได้ด้วย โดยการประกาศอาร์เรย์ด้วยชนิดของตัวแปรที่ต้องการ

```
String names[] = {"Alice", "Bob", "Charlie", "Dave"}; // ประกาศอาร์เรย์เก็บชื่อ
```

```
float temperatures[] = {23.5, 24.8, 26.1, 28.3, 27.6}; // ประกาศอาร์เรย์เก็บอุณหภูมิ
```

## 4.8 เงื่อนไข Condition

เงื่อนไข (condition) เป็นการตรวจสอบเงื่อนไขหรือเปรียบเทียบค่าของตัวแปร โดยใช้เครื่องหมายเปรียบเทียบที่น = (เท่ากับ), != (ไม่เท่ากับ), < (น้อยกว่า), > (มากกว่า), <= (น้อยกว่าหรือเท่ากับ), >= (มากกว่าหรือเท่ากับ)

ในการใช้งานเงื่อนไขใน Arduino จะใช้คำสั่ง if, else if, และ else โดยมีรูปแบบการใช้งานดังนี้

```
if (condition) {
    // กระบวนการที่จะทำเมื่อเงื่อนไขเป็นจริง
} else if (condition2) {
    // กระบวนการที่จะทำเมื่อเงื่อนไข 2 เป็นจริง
} else {
    // กระบวนการที่จะทำเมื่อไม่มีเงื่อนไขใดเป็นจริง
}
```

ตัวอย่างโปรแกรม การใช้ if (condition)

```

Condition_if
5 void led_connect()
6 {
7   digitalWrite(led_connection, HIGH);delay(1000);digitalWrite
8 }
9
10 void setup()
11 {
12   Serial.begin(9600);
13   pinMode(led_connection, OUTPUT);
14   pinMode(led_run, OUTPUT);
15 }
16
17 void loop()
18 {
19   counter++; //counter = counter + 1;
20   led_connect();
21   Serial.print("counter : ");Serial.println(counter);
22   if [counter > 10)
23   {
24     digitalWrite(led_run, HIGH);
25     Serial.println("led_run");
26   }else{
27     digitalWrite(led_run, LOW);
28   }
29 }
30

```

counter : 3  
counter : 4  
counter : 5  
counter : 6  
counter : 7  
counter : 8  
counter : 9  
counter : 10  
counter : 11  
led\_run  
counter : 12  
led\_run  
counter : 13  
led\_run  
counter : 14  
led\_run  
counter : 15  
led\_run

Autoscroll  Show timestamp

การทำงานของโปรแกรมคือ ประกาศตัวแปร int counter แล้วให้ counter บวกทีละ 1 จนค่ามากกว่า 10 โดยใช้คำสั่ง if เป็นตัวเช็คเงื่อนไข พร้อมให้ led\_run ทำงาน

โดยเงื่อนไขสามารถมีการเชื่อมต่อกันได้ด้วยเครื่องหมาย && (and) และ || (or) ดังตัวอย่างนี้

```
if (condition1 && condition2) {  
  
    // กระบวนการที่จะทำเมื่อเงื่อนไข 1 และเงื่อนไข 2 เป็นจริง
```

}

```
if (condition1 || condition2) {
```

// กระบวนการที่จะทำเมื่อเงื่อนไข 1 หรือเงื่อนไข 2 เป็นจริง

}

### ตัวอย่างโปรแกรม if(condition1 && condition2)

```

15 ,
16
17 void loop()
18{
19    counter++; //counter = counter + 1;
20    led_connect();
21    Serial.print("counter : ");Serial.println(counter);
22    if [counter > 10 && counter < 20]
23{
24        digitalWrite(led_run, HIGH);
25        Serial.println("led_run");
26    }else{
27        digitalWrite(led_run, LOW);
28    }
29}
30

```

```

counter : 6
counter : 7
counter : 8
counter : 9
counter : 10
counter : 11
led_run
counter : 12
led_run
counter : 13
led_run
counter : 14
led_run
counter : 15
led_run

```

เมื่อเพิ่มเงื่อนไขในคำสั่ง if เป็น `counter > 10 && counter < 20` ผลลัพธ์ที่ได้ `led_run` จะทำงานในช่วง `counter` มากกว่า 10 และ น้อยกว่า 20 หลังจากนั้น `led_run` หยุดการทำงาน

### ตัวอย่างโปรแกรม if(condition1 || condition2)

```

15 ,
16
17 void loop()
18{
19    counter++; //counter = counter + 1;
20    led_connect();
21    Serial.print("counter : ");Serial.println(counter);
22    if (counter > 10 || counter < 20)
23{
24        digitalWrite(led_run, HIGH);
25        Serial.println("led_run");
26    }else{
27        digitalWrite(led_run, LOW);
28    }
29}
30

```

```

counter : 14
led_run
counter : 15
led_run
counter : 16
led_run
counter : 17
led_run
counter : 18
led_run
counter : 19
led_run
counter : 20
led_run
counter : 21
led_run

```

เมื่อเพิ่มเงื่อนไขในคำสั่ง if เป็น counter > 10 || counter < 20 ผลลัพธ์ที่ได้ led\_run จะทำงานตลอด เมื่อจากเงื่อนไขกำหนดให้ counter มากกว่า 10 หรือ counter น้อยกว่า 20 จะนับได้เท่าไรก็เข้าเงื่อนไขหมด

ในการใช้งานเงื่อนไขใน Arduino จะใช้คำสั่ง while โดยมีรูปแบบการใช้งานดังนี้

คำสั่ง while ใน Arduino เป็นคำสั่งที่ใช้ในการวนลูป (loop) หรือทำงานวนซ้ำโดยตรวจสอบเงื่อนไขก่อนทำงานในแต่ละรอบ โครงสร้างคำสั่ง while มีดังนี้

while (ເງື່ອນໄຂ)

{

## // ทำงานที่ต้องการวนลูป

}

ในการทำงาน โปรแกรมจะตรวจสอบเงื่อนไขก่อนทำงานในแต่ละรอบของลูป หากเงื่อนไขเป็นจริง (true) โปรแกรมจะทำงานภายในลูป และวนลูปไปเรื่อยๆ จนกว่าเงื่อนไขจะเป็นเท็จ (false) แล้วจึงออกจากลูป

ตัวอย่างโปรแกรม คำสั่ง while

เมื่อเงื่อนไขในคำสั่ง while เป็น counter > 10 ผลลัพธ์ที่ได้ led\_run จะทำงานตลอด เมื่อ counter มากรกว่า 10 จนนั้นจะไม่มีการนับใดเนื่องจากการทำงานของโปรแกรมจะทำงานในลูป ของ while(counter > 10) { } เท่านั้น

ในการใช้งานเงื่อนไขใน Arduino จะใช้คำสั่ง for โดยมีรูปแบบการใช้งานดังนี้

คำสั่ง for ใน Arduino เป็นคำสั่งที่ใช้ในการวนลูป (loop) หรือทำงานวนซ้ำโดยกำหนดจำนวนรอบการทำงานล่วงหน้า โครงสร้างของคำสั่ง for มีดังนี้

for (เงื่อนไขเริ่มต้น; เงื่อนไขสิ้นสุด; การเปลี่ยนแปลงค่าตัวแปร ในแต่ละรอบ)

{

// ทำงานที่ต้องการวนลูป

}

ในการทำงาน โปรแกรมจะเริ่มต้นทำงานภายในลูป โดยกำหนดเงื่อนไขเริ่มต้น และตรวจสอบเงื่อนไขสิ้นสุดก่อนทำงานในแต่ละรอบ หากเงื่อนไขเป็นจริง (true) โปรแกรมจะทำงานภายในลูป และดำเนินการเปลี่ยนแปลงค่าตัวแปรตามการเปลี่ยนแปลงที่กำหนด จนกว่าเงื่อนไขสิ้นสุดจะเป็นเท็จ (false) และจึงออกจากลูป

ตัวอย่างโปรแกรม คำสั่ง for

```

3
10 void setup()
11 {
12   Serial.begin(9600);
13   pinMode(LED_CONNECTION, OUTPUT);
14   pinMode(LED_RUN, OUTPUT);
15 }
16
17 void loop()
18 {
19   counter++; //counter = counter + 1;
20   led_connect();
21   Serial.print("counter : ");Serial.println(counter);
22   for (int i=0;i<=10;i++)
23   {
24     digitalWrite(LED_RUN, HIGH);
25     Serial.print(i);Serial.println(" led_run");
26   }
27   digitalWrite(LED_RUN, LOW);
28 }
29

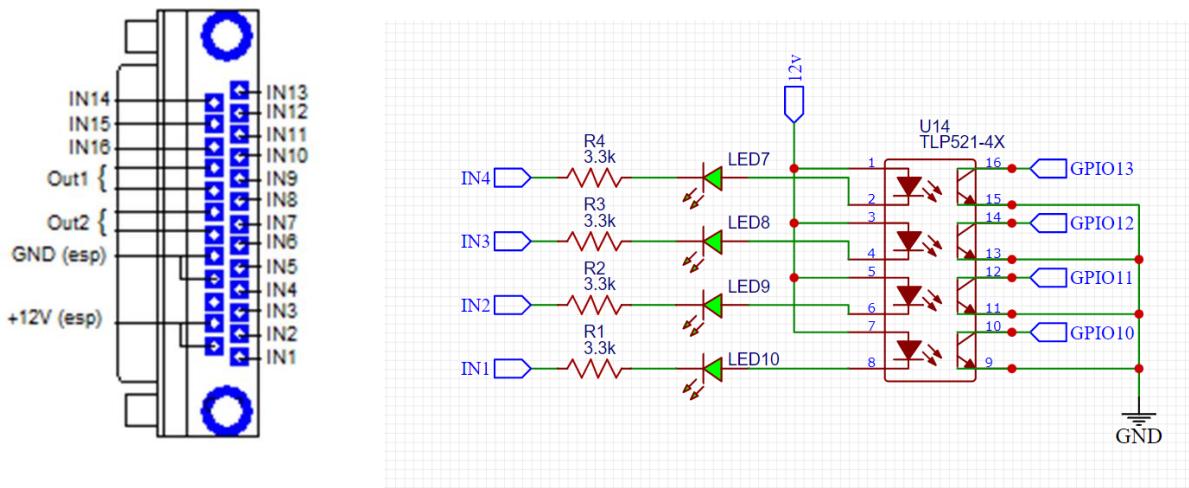
```

Done uploading.

คำสั่ง for กำหนดค่าเริ่มต้น  $i=0$ ; เนื่องไปสื้นสุด  $i \leq 10$ ; ค่าเปลี่ยนแปลง  $i++$ ; ผลลัพท์ที่ได้คือ led\_run จะทำงานทุกครั้งต่อรอบแต่จะทำงานภายในเงื่อนไข  $i \leq 10$  เท่านั้นแล้วออกจากลูป

#### 4.9 Digital Input & Digital Output

Digital Input คือ สัญญาณแบบดิจิตอลที่เข้าสู่อุปกรณ์หรือระบบจากแหล่งข้อมูลภายนอก เช่น สัญญาณจากเซ็นเซอร์ สัญญาณจากสวิตช์ หรือสัญญาณจากอุปกรณ์อื่น ๆ โดยสัญญาณดิจิตอลจะมีสถานะเป็น 0 หรือ 1 เท่านั้น หรืออาจระบุเป็นเลขฐานสอง เช่น 0V หรือ 3.3V สำหรับระบบดิจิตอล โดยอุปกรณ์หรือระบบต่าง ๆ จะนำข้อมูลนี้ไปประมวลผลหรือใช้งานต่อไปตามที่ต้องการ เช่น การควบคุมอุปกรณ์ต่าง ๆ หรือการแสดงผลข้อมูลบนหน้าจอ หรือการทำงานอื่น ๆ ที่เกี่ยวข้อง กับการรับสัญญาณนำเข้าแบบดิจิตอล



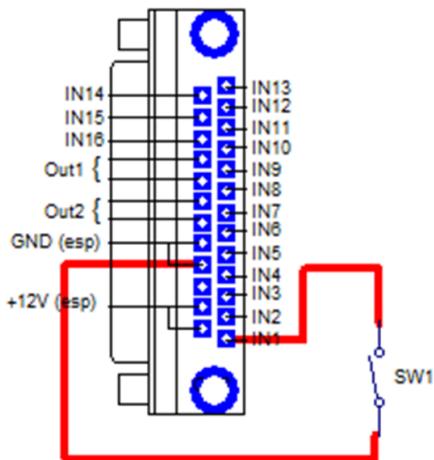
แสดงการต่อวงจรสั่งงาน Digital Input

PIN	GPIO
IN1	10
IN2	11
IN3	12
IN4	13
IN5	14
IN6	15
IN7	16
IN8	21
IN9	40
IN10	39
IN11	38
IN12	37
IN13	36
IN14	35
IN15	34
IN16	33

PIN	GPIO
Out1	3
Out2	1

แสดงจำนวน GPIO ทั้งหมดที่ใช้งาน



สถานะปกติ = HIGH หรือ 1

เมื่อสวิตซ์ถูกกด = LOW หรือ 0

แสดงการวิธีสั่ง Digital Input ทำงาน ให้ทำการนำขา IN ที่ต้องการไปเสียบที่ขา GND

### ตัวอย่างโปรแกรม Digital Input

#### Digital\_Input §

```

1 #define in1 10
2 #define in2 11
3
4 #define led_run 6
5
6 int buttonState1 = 0;
7 int buttonState2 = 0;
8
9 void setup()
10 {
11   pinMode(in1, INPUT);
12   pinMode(in2, INPUT);
13   pinMode(led_run, OUTPUT);
14 }
15
16 void loop()
17 {
18   buttonState1 = digitalRead(in1);
19   buttonState2 = digitalRead(in2);
20
21   if (buttonState1 == LOW || buttonState2 == LOW)
22   {
23     digitalWrite(led_run, HIGH);
24   } else
25   {
26     digitalWrite(led_run, LOW);
27   }
28 }
```

#### กำหนดให้

In1 ต้องอยู่กับ pin GPIO10 เป็น Input

In2 ต้องอยู่กับ pin GPIO11 เป็น Input

สถานะเริ่มต้น pin GPIO10,GPIO11 จะเท่ากับ HIGH เมื่อมีการกดสวิตช์ in1 หรือ in2 ก็จะทำให้หลอด led\_run ทำงาน เมื่อปล่อยสวิตช์หลอด led\_run หยุดทำงาน

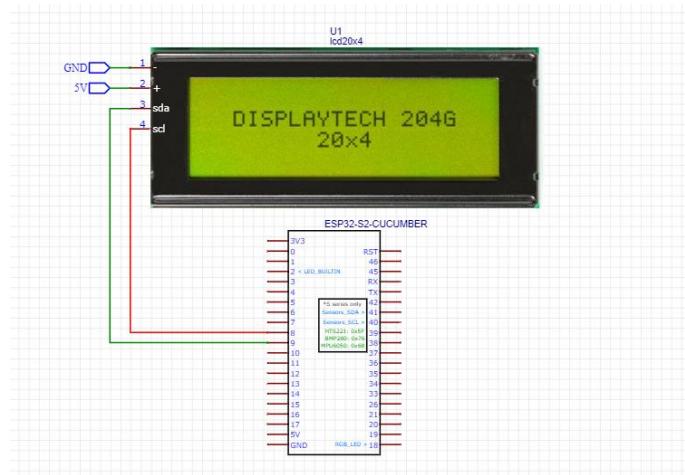
#### 4.10 LCD

LCD (Liquid Crystal Display) คือ หน้าจอแสดงผล โดยมีการควบคุมภาพแสดงผลโดยวิธีการเปลี่ยนแปลงทิศทางของแสง และความโปร่งแสงของแต่ละส่วนบนหน้าจอ ซึ่ง LCD มีความเหมาะสมสำหรับการใช้งานในอุตสาหกรรม และผู้ใช้ทั่วไป เนื่องจากมีขนาดเล็ก น้ำหนักเบา และประหยัดพลังงาน และใช้กันอย่างแพร่หลายในอุปกรณ์ต่างๆ เช่น โทรศัพท์มือถือ แท็บเล็ต กล้องดิจิตอล รถยนต์ และอื่นๆ



รูปตัวอย่าง LCD 20x4

การเชื่อมต่อแบบ I2C (Inter-Integrated Circuit) ระหว่างหน้าจอแสดงผล LCD (Liquid Crystal Display) กับไมโครคอนโทรลเลอร์ (Microcontroller) โดยใช้ชิปแปลงสัญญาณ (Converter Chip) ที่ช่วยเชื่อมต่อระหว่าง I2C และ LCD ให้ได้ โดยเป็นวิธีการเชื่อมต่อที่สะดวกและประหยัดพื้นที่ เนื่องจาก I2C ใช้สายสัญญาณเพียงสองเส้น (SDA และ SCL) เท่านั้น ชาที่ใช้งาน SDA ให้ต่ออยู่กับ GPIO8 ส่วน SCL ต่ออยู่กับขา GPIO9 ตามรูป



แสดงการต่อวงจรสั่งงาน LCD20x4 with I2C interface

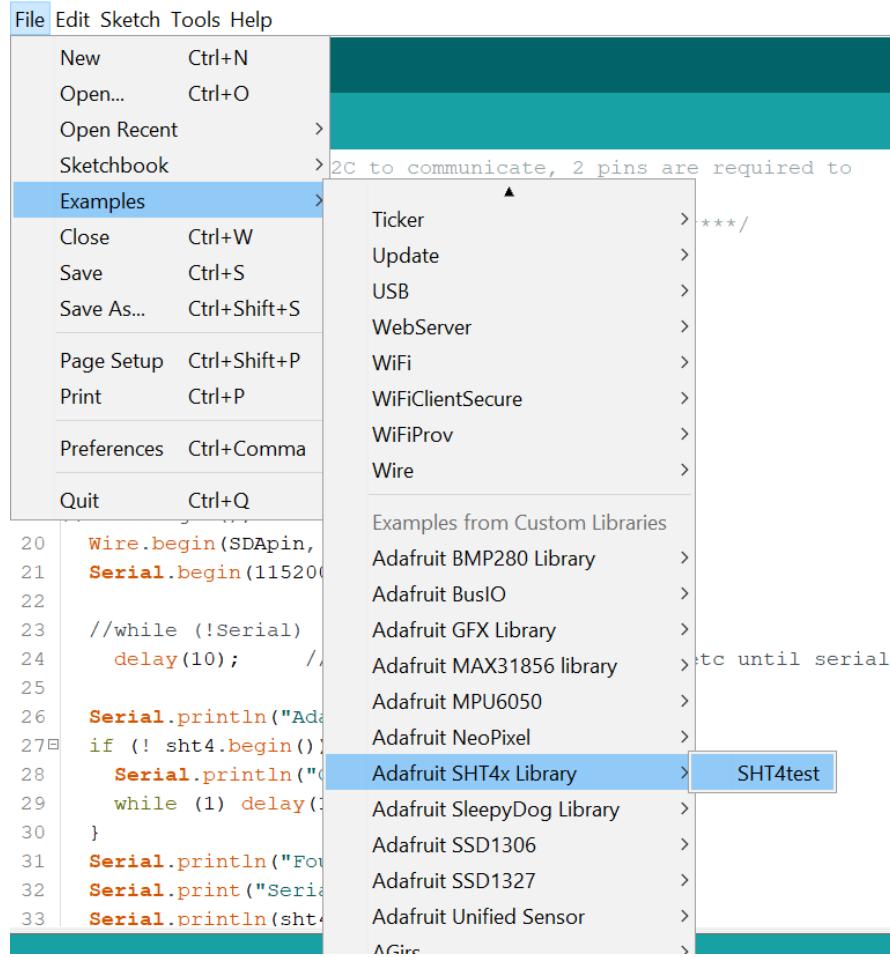
## ตัวอย่างการเขียนโปรแกรม LCD20x4 with I2C interface

LCD	เรียกใช้งาน Library Wire.h, LiquidCrystal_I2C.h
1 #include <Wire.h>	LiquidCrystal_I2C lcd(0x27, 20, 4); คือการกำหนด
2 #include <LiquidCrystal_I2C.h>	Address : 0x27 ขนาดหน้าจอ 20 คอลั่ม 4 แถว
3	
4 LiquidCrystal_I2C lcd(0x27, 20, 4);	
5	
6 void setup()	เรียกใช้งานฟังก์ชัน lcd.begin();
7 {	
8 lcd.begin();	Lcd.setCursor(0, 0); กำหนดจุดเริ่มต้นตัวอักษร
9	
10 }	
11	
12 void loop()	Lcd.print("MIC Division"); สั่งให้หน้าจอแสดงคำว่า
13 {	MIC Division
14 lcd.setCursor(0, 0);	
15 lcd.print("MIC Division");	
16 }	

### 4.11 Sensor SHT4X

เซ็นเซอร์ SHT4X เป็นเซ็นเซอร์วัดอุณหภูมิและความชื้นในอากาศที่มีการติดต่อผ่าน I2C (Inter-Integrated Circuit) ซึ่งเป็นหนึ่งในโปรโตคอลสื่อสารระหว่างไมโครคอนโทรลเลอร์ โดย SHT4X มีความแม่นยำสูงในการวัดและเป็นเซ็นเซอร์ที่มีราคาไม่แพง เหมาะสมสำหรับใช้ในการวัดอุณหภูมิและความชื้นในโครงงาน Arduino หรือ IoT ต่างๆ

การติดต่อกับ SHT4X ผ่าน I2C สามารถทำได้ด้วยการเชื่อมต่อขา SDA และ SCL ของโมดูลกับขา SDA(GPIO41) และ SCL(GPIO40) ของบอร์ด ESP32S2 ตามลำดับ โดยสามารถใช้ไลบรารี Wire ของ Arduino และต้องติดตั้ง Libeary Adafruit\_SHT4x.h , Adafruit\_Busio, Adafruit Unified Sensor ในการสื่อสารกับ SHT4X ได้ดังนี้



### เลือกตัวอย่าง SHT4test

```

10 #include <Wire.h>
11
12 #include "Adafruit_SHT4x.h"
13
14 const int SCLpin = 40;
15 const int SDApin = 41;
16
17 Adafruit_SHT4x sht4 = Adafruit_SHT4x();
18
19 void setup() {
20
21     Wire.begin(SDApin, SCLpin);
22
23     Serial.begin(115200);
24
25     //while (!Serial)
26     //    delay(10);      // will pause Zero, Leonardo
27
28     Serial.println("Adafruit SHT4x test");
29     if (! sht4.begin()) {

```

COM54

```

Read duration (ms): 11
Temperature: 27.01 degrees C
Humidity: 65.35% rH
Read duration (ms): 11
Temperature: 27.03 degrees C
Humidity: 65.30% rH
Read duration (ms): 11
Temperature: 27.03 degrees C
Humidity: 65.24% rH
Read duration (ms): 11
Temperature: 27.05 degrees C
Humidity: 65.19% rH
Read duration (ms): 11
Temperature: 27.05 degrees C
Humidity: 65.16% rH
Read duration (ms): 11

```

ให้ทำการเพิ่มโค๊ดในการอ่านสีเหลี่ยม คือการติดต่อผ่าน I2C (Inter-Integrated Circuit) กำหนดให้ SDA เป็นขา GPIO40 และ SCL เป็นขา GPIO41 ผลลัพท์จะได้ค่าอุณหภูมิความชื้นและเวลาของรอบการทำงานใน Serial monitor

#### 4.12 Watch Dog Timer

Watchdog Timer หรือ WDT เป็นคำสั่งที่มักนำมาใช้ในการควบคุมความเสี่ยงในการทำงานของระบบในโครคอน โทรเลอร์หรือ MCU โดยเฉพาะอย่างยิ่งในการทำงานของระบบอัตโนมัติหรือระบบที่ต้องการความเสถียรสูง โดย WDT จะทำหน้าที่ตรวจสอบการทำงานของระบบและรีเซ็ตระบบเมื่อเกิดข้อผิดพลาดหรือติดอยู่ในสถานะไม่สมบูรณ์

WDT ทำงานโดยตรวจสอบเงื่อนไขเวลาที่กำหนดเอง โดยมักตั้งค่าให้ WDT เป็นเวลาสั้น ๆ เพียงพอที่จะเช็คและตรวจสอบการทำงานของระบบอยู่เสมอ ถ้า WDT ไม่ได้รับสัญญาณจากระบบภายในเวลาที่กำหนด จะมีการเรียกใช้คำสั่งรีเซ็ต (reset) เพื่อรีเซ็ตระบบไปยังสถานะเริ่มต้น เพื่อป้องกันไม่ให้ระบบเข้าสู่สถานะที่ผิดปกติที่อาจทำให้เกิดความเสียหายแก่การทำงานของระบบ

การเขียนโค๊ด WDT ใน Arduino สามารถทำได้โดยใช้ไลบรารี (library) ที่มากับ Arduino IDE ซึ่งมีฟังก์ชันเรียกใช้ WDT อยู่แล้ว ดังนี้

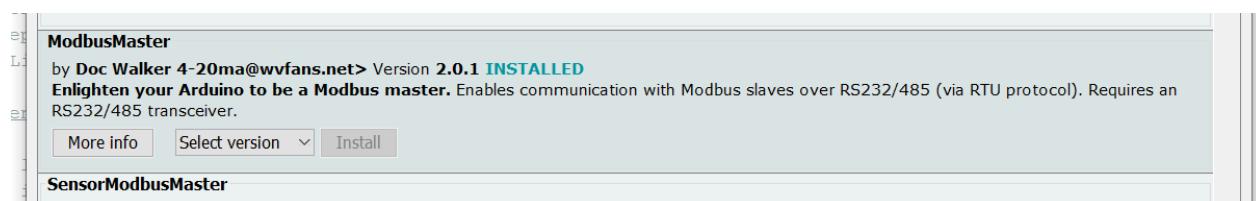
```
1 #include <esp_task_wdt.h>
2 #define WDT_TIMEOUT 30
3
4 void setup()
5 {
6     Serial.begin(115200);
7     esp_task_wdt_init(WDT_TIMEOUT, true);
8     esp_task_wdt_add(NULL);
9     esp_task_wdt_reset();
10    delay(1000);
11    Serial.println("ESP_RESET");
12    delay(1000);
13 }
14
15 void loop()
16 {
17     Serial.println("ESP_RUN");
18     for(int i=0;i<=10;i++)
19     {
20         Serial.print("i :");Serial.println(i);
21         delay(1000);
22     }
23     esp_task_wdt_reset();
24     delay(1000);
25 }
```

#### 4.13 Modbus Master

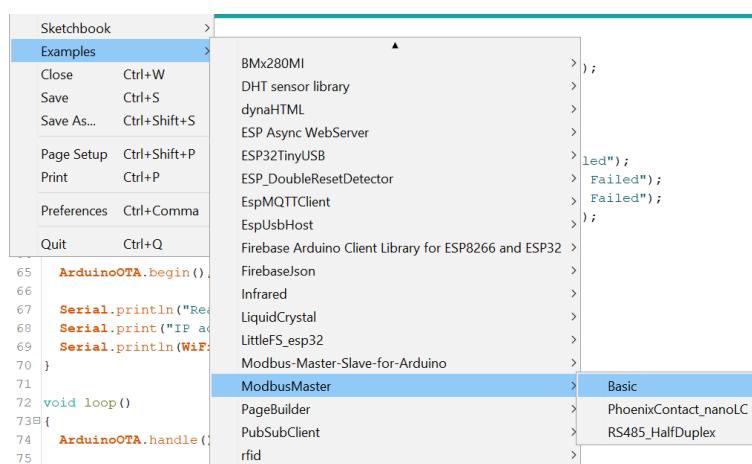
Modbus เป็นโปรโตคอลการสื่อสารที่ได้รับความนิยมในระบบอุตสาหกรรมเพื่อการสื่อสารระหว่างอุปกรณ์ต่างๆ เช่น PLC, HMI เป็นต้น ซึ่งได้อธิบายดังบทเรียนที่ผ่านมา

ในตัวอย่างนี้จะทำการอ่านค่า ReadHoldingRegisters จากโปรแกรม Modbus Slave มีขั้นตอนดังต่อไปนี้

- ติดตั้ง Library ModbusMaster.h



- เมื่อติดตั้ง Library ModbusMaster เสร็จเรียบร้อยแล้ว ให้ไปที่ Examples ที่มีชื่อว่า Basic



- ให้แก้ไขโค้ดกำหนดให้ Serial.begin(9600); Serial1.begin(9600); และ Slave ID

เท่ากับ 1 ในที่นี่เราจะติดต่อสื่อสาร Slave ID 1 ที่ port Serial1 9600 baud

```

30
31 void setup()
32 {
33     // use Serial (port 0); initialize Mc
34     Serial.begin(9600);
35     Serial1.begin(9600);
36
37     // communicate with Modbus slave ID 2
38     node.begin(1, Serial1);
39 }

```

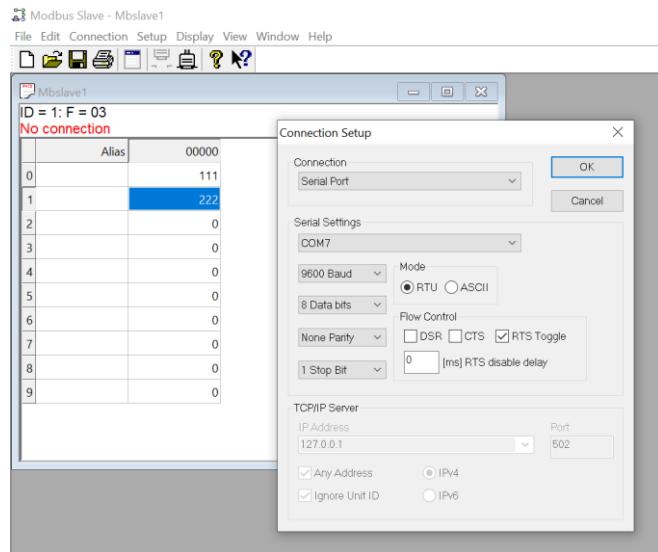
4. ทำการลบโค้ดที่ไม่ได้ใช้งานและแก้ result = node.readHoldingRegisters(0, 6); จะทำ  
การอ่าน registers ตัวที่ 0 – 6 เมื่อ result == node.ku8MBSuccess ก็ทำการ  
node.getResponseBuffer(); มาเก็บไว้ในตัวแปร data[j] จากนั้น Upload Program

```

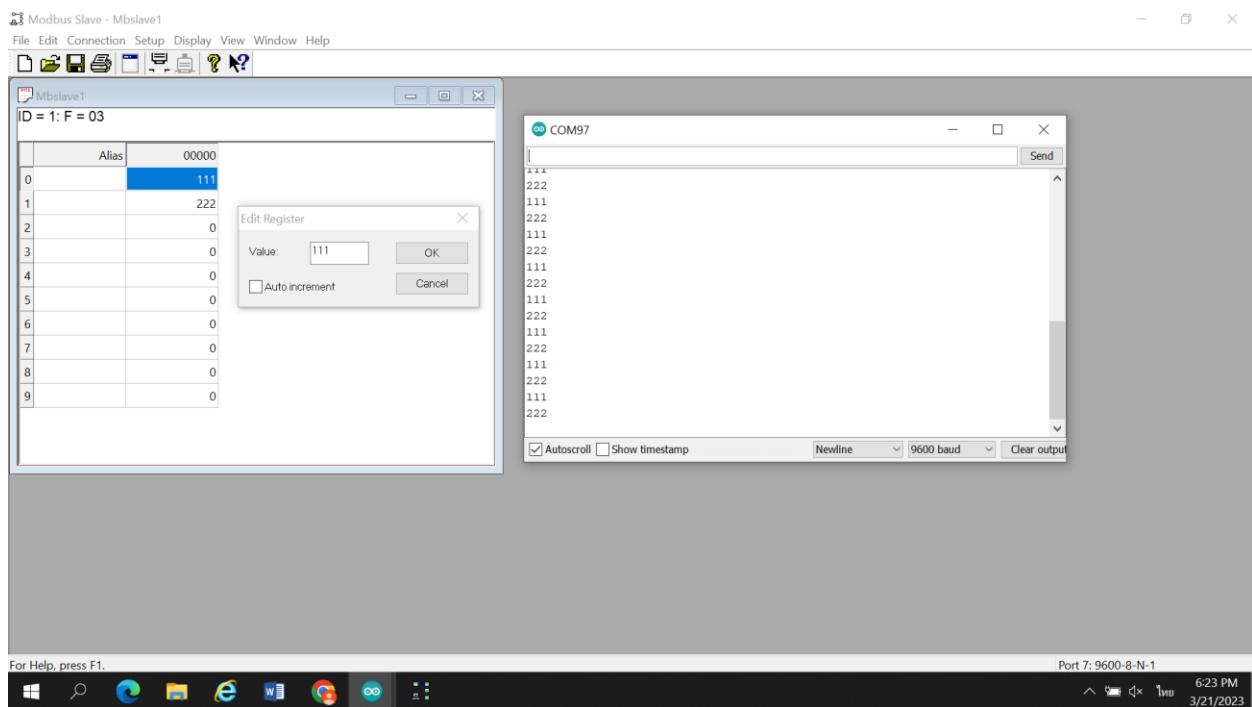
41 void loop()
42 {
43     static uint32_t i;
44     uint8_t j, result;
45     uint16_t data[6];
46
47     result = node.readHoldingRegisters(0, 6);
48
49     if (result == node.ku8MBSuccess)
50     {
51         for (j = 0; j <= 1; j++)
52         {
53             data[j] = node.getResponseBuffer(j);
54             Serial.println(data[j]);
55             delay(5000);
56         }

```

5. Setting โปรแกรม Modbus slave จากนั้นนำสาย RS232 to USB Cable มาเสียบที่  
คอมพิวเตอร์ เข้าไปที่ Connection setup เลือก Com Port และ 9600 Baud ดังรูป



6. ให้ทำการ Edit Register ที่ต้องการแล้วกด OK ก็จะมีค่าโชว์ใน Serial moniter



#### 4.14 Modbus Slave

ในหัวข้อนี้กำหนดให้กล่อง TRC-001A เป็น Slave และทำการอ่านค่า ReadHoldingRegisters จากโปรแกรม Modbus Poll มีขั้นตอนดังต่อไปนี้

#### ตัวอย่างการเขียนโปรแกรม ModbusSlave

Download <https://github.com/golf1481/TRC001> เมื่อโหลดเสร็จให้นำไฟล์ Modbus\_slave.ino และ ModbusRTU.h ไว้ในโฟลเดอร์เดียวกัน

```

ModbusSlave ModbusRTU.h
1 #include "ModbusRtu.h"
2 #define ID 1
3
4 Modbus slave(ID, Serial1, 0);
5
6 int8_t state = 0;
7
8 int num = 2;
9
10 uint16_t au16data[20];
11
12 void setup()
13 {
14     pinMode(42, OUTPUT);
15     Serial.begin(9600);
16     Serial1.begin(9600);
17     slave.start();
18 }
19
20 void loop()
21 {
22     state = slave.poll( au16data ,num );
23
24     if (state == 7) {
25         Serial.print("----Waiting----");
26     }
27
28     if (state == 8)
29     {
30         Serial.println("----START ----");
31         for (int i = 0; i < (num * 2); i++)
32         {
33             slave.poll( au16data, num );
34             delay(100);
35         }
36
37         digitalWrite(42, LOW );
38         Serial.print("1.D200 : "); Serial.println(au16data[0]);
39         Serial.print("2.D201 : "); Serial.println(au16data[1]);
40
41         Serial.println("-----");
42         Serial.println(" ");
43         delay(5000);
44     }
45 }
46 
```

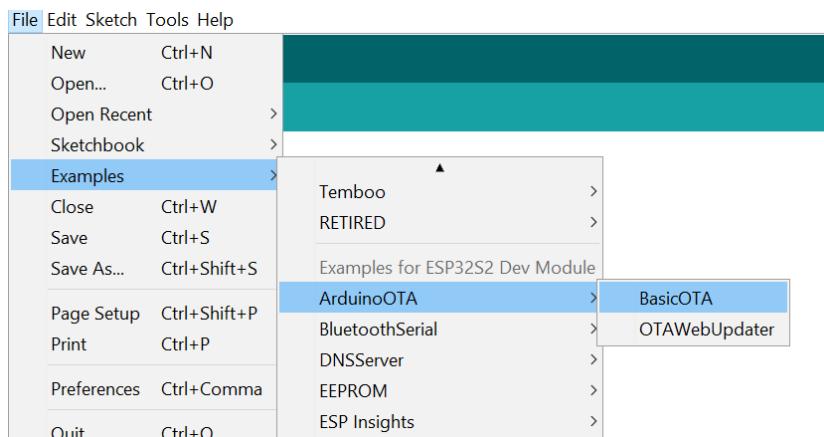
- `#include "ModbusRTU.h"` เป็นการเรียก library ModbusRTU.h
- `#define ID 1` กำหนดให้ ID มีค่าเป็น 1
- `Modbus slave(ID, Serial1, 0);` กำหนดค่าในฟังก์ชัน Modbus slave (Address, Port ที่อ่าน, RS232 = 0 หรือ RS485 = 1)
- `int num = 2;` ประกาศตัวแปร num เป็น int เพื่อกำหนดจำนวน Register ในการอ่านค่า

- slave.start(); พังก์ชั่นเริ่มการทำงาน
- state = slave.poll( au16data ,num ); สแกนหาสถานะการต่อสารภัยบอร์ด
- state == 7 สถานะรอการตอบกลับจากอุปกรณ์
- state == 8 สถานะติดต่ออุปกรณ์ที่เป็น Master ได้สำเร็จ จากนั้นใช้คำสั่ง for ในการวนอ่านค่ามาเก็บไว้ในตัวแปร au16data

ในการทดลองติดต่อสารภัยบอร์ดที่เป็น Master โดยใช้โปรแกรม Modbus poll

#### 4.15 OTA Static IP

OTA (Over-the-Air) หมายถึงการอัปโหลดโปรแกรมลงบอร์ด Arduino โดยตรงผ่านทางเครือข่าย WiFi โดยไม่ต้องเชื่อมต่อผ่านสาย USB หรืออุปกรณ์อื่นๆ เพื่อให้การอัปโหลดโปรแกรมสะดวกสบายมากยิ่งขึ้น สามารถดูตัวอย่าง OTA Basic ได้ดังรูป



จากนั้นให้ทำการใส่ ssid, password

```

1 #include <WiFi.h>
2 #include <ESPmDNS.h>
3 #include <WiFiUdp.h>
4 #include <ArduinoOTA.h>
5
6 const char* ssid = ".....";
7 const char* password = ".....";
8

```

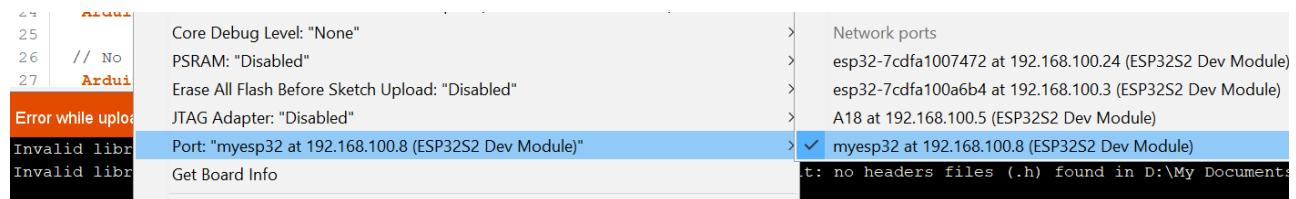
ตั้งชื่อ Hostname (ชื่อของกล่อง) และ Password เสร็จแล้วให้ Upload Program

```

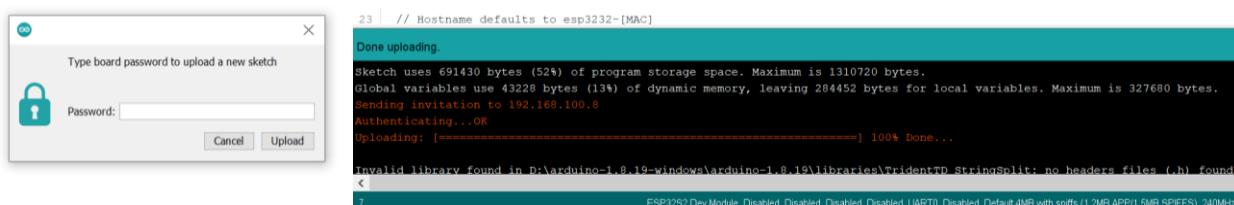
23 // Hostname defaults to esp3232-[MAC]
24 ArduinoOTA.setHostname("myesp32");
25
26 // No authentication by default
27 ArduinoOTA.setPassword("admin");
28

```

เมื่อ Upload Program ผ่านสาย USB เสร็จแล้วให้สังเกตุที่ Port จะปรากฏชื่อ Hostname , IP Address และ Dev Module จากนั้นให้ทำการเลือกแล้ว Upload Program



ใส่ Password แล้วกด Upload



ในการใช้งาน OTA บน Arduino โดยใช้ IP แบบคงที่ (Static IP) จะต้องทำการกำหนด IP ให้กับบอร์ด Arduino โดยตรง โดยสามารถทำได้โดยการใช้โค้ดตัวอย่างดังนี้

```

6 | const char* ssid = ".....";
7 | const char* password = ".....";
8 |
9 | IPAddress local_IP(192, 168, 1, 28); // Static IP
10| IPAddress gateway(192, 168, 1, 1); // Gateway
11| IPAddress subnet(255, 255, 255, 0); // subnet
12|
13|
14void setup() {
15  Serial.begin(115200);
16  Serial.println("Booting");
17  //WiFi.mode(WIFI_STA);
18  WiFi.config(local_IP, gateway, subnet);
19  WiFi.begin(ssid, password);
20  while (WiFi.waitForConnectResult() != WL_CONNECTED) {
21    Serial.println("Connection Failed! Rebooting...");
22    delay(5000);
23    ESP.restart();
24  }
25}

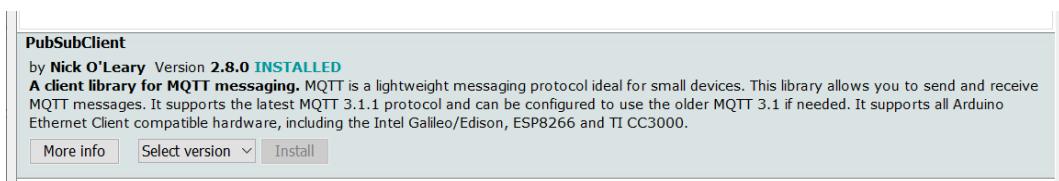
```

เพิ่มแก้ไขโค้ดในกรอบสีเหลี่ยม ให้ทำการกำหนด local\_IP, geteway, subnet และทำการ Upload Program ผ่าน OTA เมื่ออัพโหลดเสร็จจะสังเกตุว่าที่ local\_IP จะเปลี่ยนตามที่เรากำหนด คุณได้จาก Serial monitor

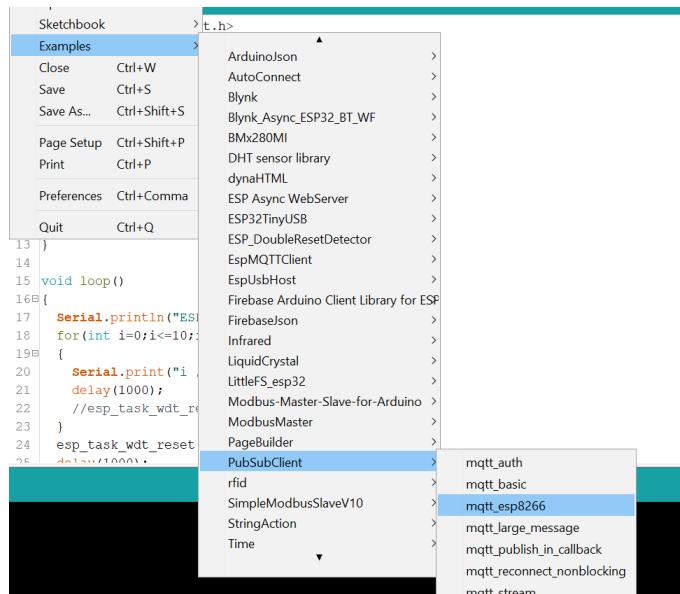
#### 4.16 PubSubClient

PubSubClient คือ library (หรือ ไลบรารี) ของภาษา Arduino ที่ช่วยให้เราสามารถเชื่อมต่อ และสื่อสารกับ MQTT broker ได้อย่างง่ายดาย โดย MQTT (Message Queuing Telemetry Transport) เป็นโปรโตคอลสื่อสารที่ออกแบบมาเพื่อใช้ในการสื่อสารข้อมูลในรูปแบบ publish/subscribe โดย PubSubClient ช่วยให้เราสามารถส่งข้อมูล (publish) หรือรับข้อมูล (subscribe) ผ่าน MQTT broker ได้โดยไม่จำเป็นต้องเขียนโค้ด MQTT ขึ้นมาเอง ซึ่งทำให้การเชื่อมต่อและสื่อสารกับอุปกรณ์ IoT ที่ใช้ MQTT เป็นเรื่องง่ายขึ้นมากขึ้นกว่าการเขียนโค้ด MQTT ขึ้นมาเองทั่วไป มีขั้นตอนดังนี้

##### 1. ติดตั้ง Library PubSubClient.h



2. เมื่อติดตั้ง Library PubSubClient เสร็จเรียบร้อยแล้ว ให้ไปที่ Examples ที่มีชื่อว่า Basic



3. !! กด `#include<ESP8266WiFi.h>` เป็น `#include<WiFi.h>` ใส่ ssid, password และ mqtt\_server (IP Address ของเครื่อง Server )

```

21 #include <WiFi.h>
22 #include <PubSubClient.h>
23
24 // Update these with values suitable for your network.
25
26 const char* ssid = ".....";
27 const char* password = ".....";
28 const char* mqtt_server = "broker.mqtt-dashboard.com";
29

```

4. เพิ่ม โค้ด led\_connection ในการแสดงสถานะการติดต่อ WiFi และแสดงสถานะการติดต่อ SERVER

```

26
27 #define led_connection 5
28 #define led_run6
29
30 int count_connection;
31
32 void setup_wifi() {

```

แก้ไขโค้ดในฟังก์ชัน void setup\_wifi()

```

42 WiFi.begin(ssid, password);
43
44 while (WiFi.status() != WL_CONNECTED) {
45   count_connection++;
46   digitalWrite(led_connection, HIGH);delay(100);digitalWrite(led_connection, LOW);delay(100);
47   Serial.print(".");
48   if(count_connection>20)
49   {
50     ESP.restart();
51   }
52 }
53
54 randomSeed(micros());

```

ประกาศตัวแปร int time\_count;

```

80 }
81
82 int time_count;
83
84 void reconnect() {
85   // Tries to connect every 5 seconds

```

แก้ไขโค้ดในฟังก์ชัน void reconnect()

```

102 // Wait 5 seconds before retrying
103 //delay(5000);
104 time_count++;
105 digitalWrite(led_connection, HIGH);delay(500);digitalWrite(led_connection, LOW);delay(500);
106 if(time_count>=10)
107 {
108   ESP.restart();
109 }
110 }
111 }
112 }
113

```

5. Upload program แล้วทดสอบการติดต่อ WiFi เพื่อคุณสามารถทำงานของ

led\_connection

6. แก้ไขโค้ดตามรูปด้านล่าง เพื่อทดสอบ publish data ที่มีชื่อ outTopic

```

127 int counter;
128 String d200;
129 char d0[16];
130
131 void loop() {
132
133 if (!client.connected()) {
134     reconnect();
135 }
136 client.loop();
137 counter++;
138 delay(1000);
139 d200 = String(counter);
140 d200.toCharArray(d0, 16);client.publish("outTopic", d0);
141 }
```

\* โค้ดที่ได้จะนำไปใช้ในหัวข้อ 5.2 วิธีการ Publish ข้อมูลจาก MIC SMART ไปที่ Node-red

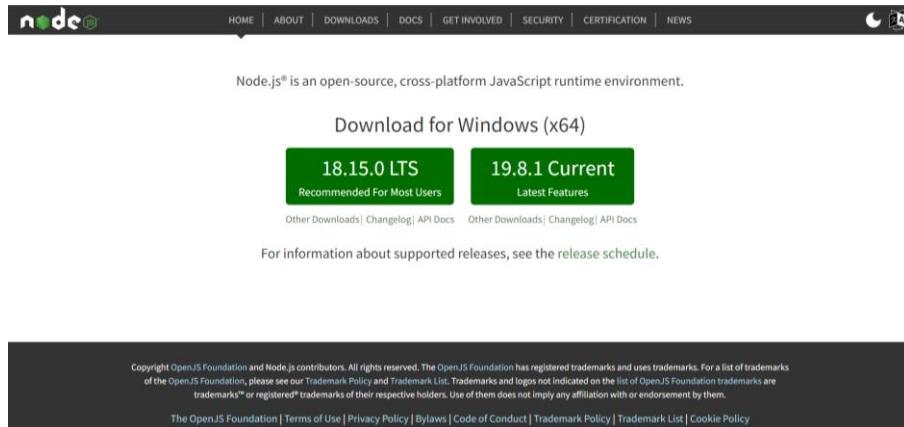
## CHAPTER 5: MQTT Protocol

---

### 5.1 Node-red Structure

#### ขั้นตอนการติดตั้ง Node-red

1. Install Node JS - <https://nodejs.org/en/>



2. Install Node RED on Windows - `npm install -g --unsafe-perm node-red`

```
c:\ npm install node-red
Microsoft Windows [Version 10.0.19044.2604]
(c) Microsoft Corporation. All rights reserved.

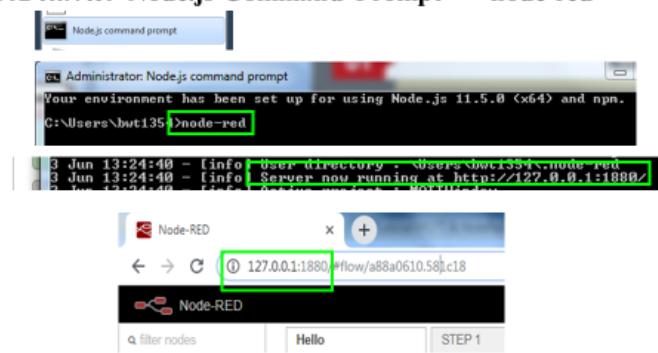
C:\Users\lblc879> npm install -g --unsafe-perm node-red
'-' is not recognized as an internal or external command,
operable program or batch file.

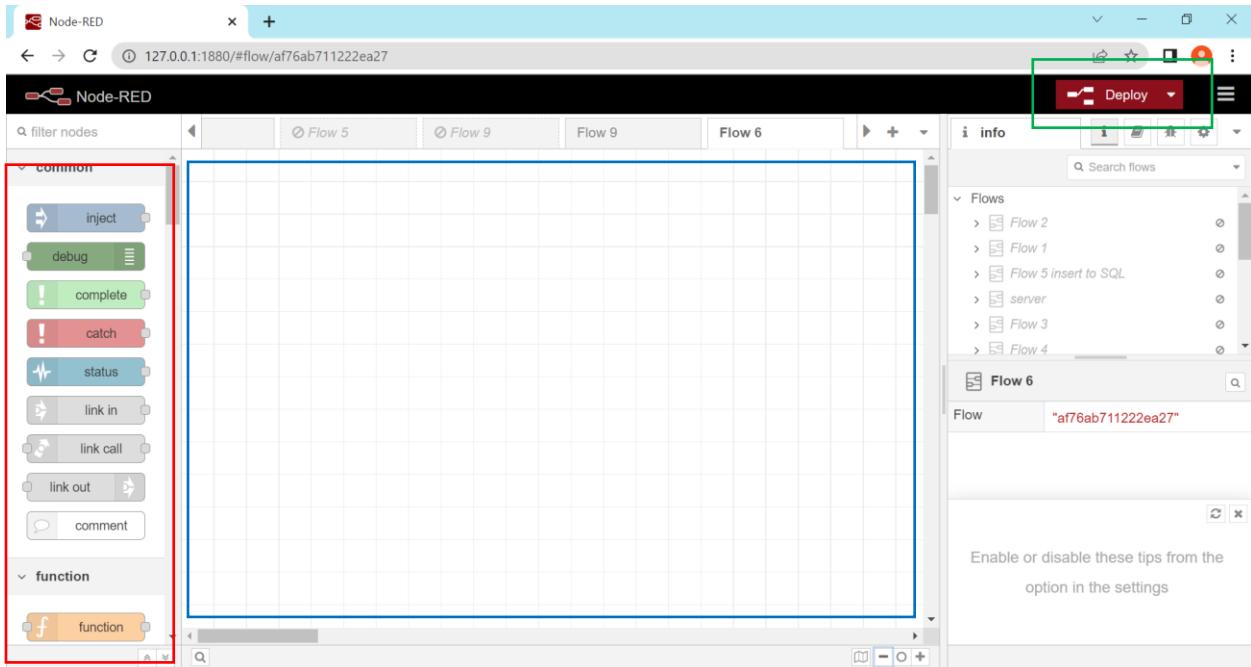
C:\Users\lblc879>npm install -g --unsafe-perm node-red
npm WARN config global '-global', '--local' are deprecated. Use `--location=global` instead.
[redacted] \ reify:@node-red/runtime: http fetch GET 200 https://registry.npmjs.org/@node-red/runtime/-/runti
```

#### ขั้นตอน Start Run Node-Red Server

##### Start Run Node-Red Server

เข้าโปรแกรม Node.js Command Prompt >> node-red



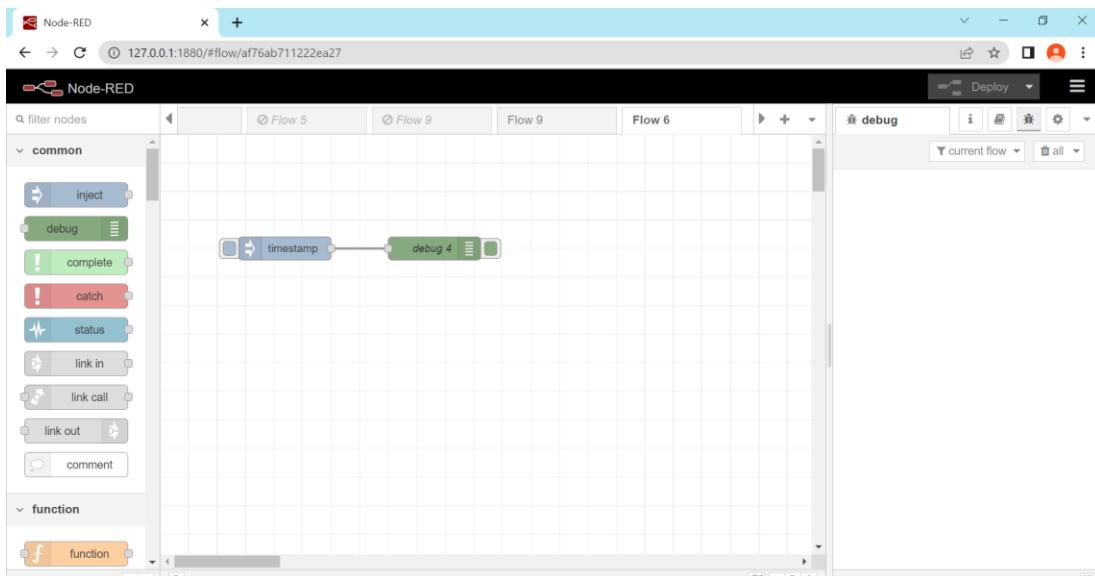


## โครงสร้างหลักของ Node red ประกอบด้วย 3 ส่วน

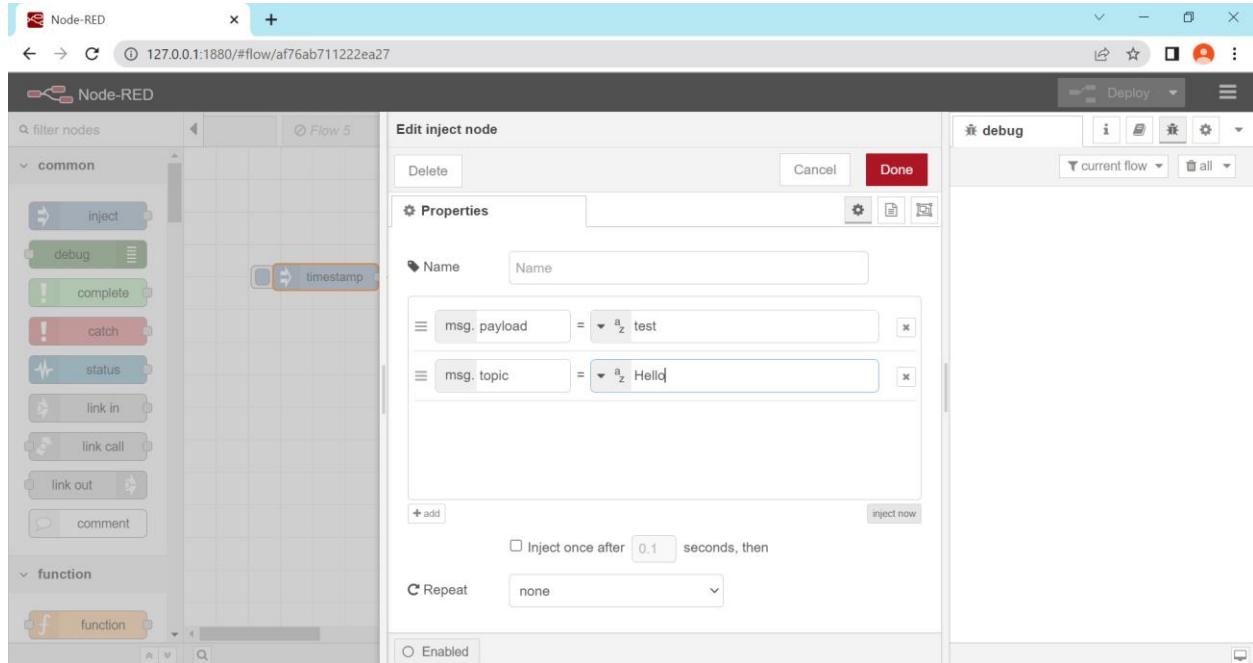
1. **Flow** พื้นที่ทำงาน
2. **Node** คำสั่งต่างๆที่สามารถใช้สร้างเงื่อนไขใน Flow
3. **Deploy** กดทุกครั้งเมื่อมีการแก้ไข Flow

ตัวอย่างในการสร้าง Flow เมื่อต้นในการแสดงข้อมูล debug

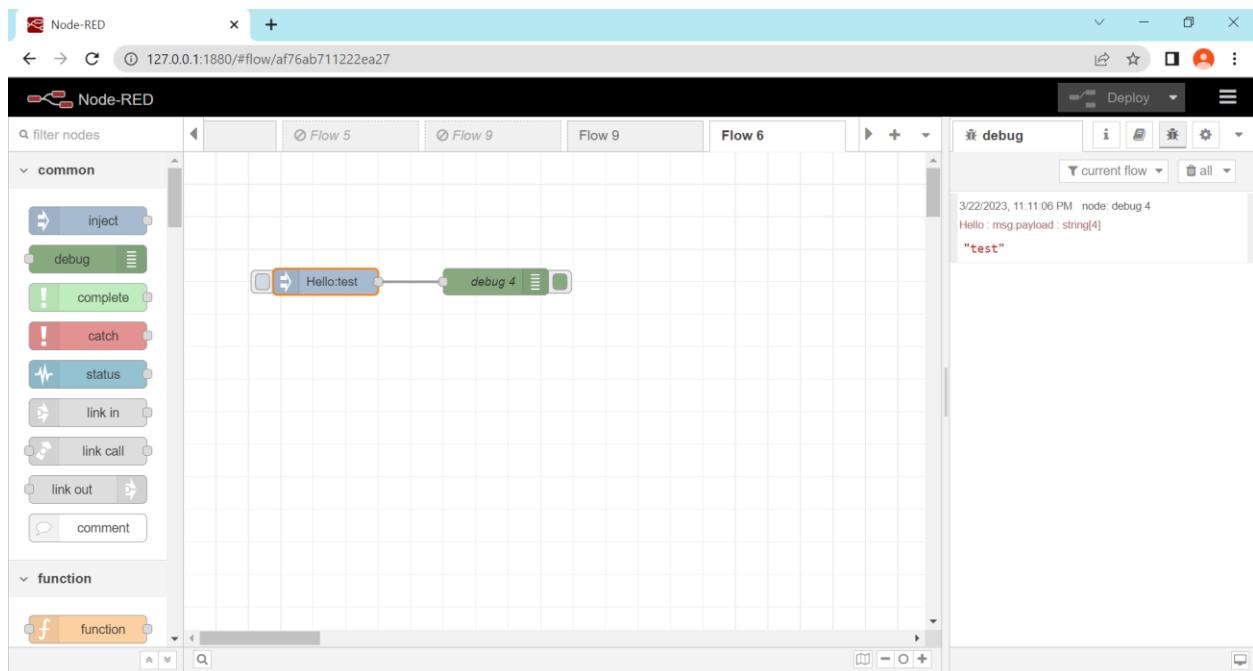
ให้ทำการลาก inject กับ debug วางบน Flow และเชื่อมต่อกัน



ตั้งค่า inject โดยกำหนด topic “Hello” และ payload “test” จากนั้นกด Done



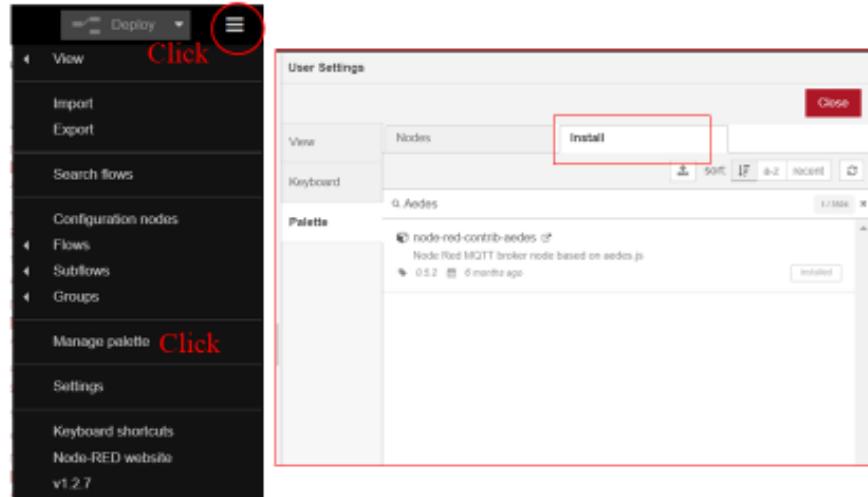
กด deploy ทดสอบ Flow โดยกด inject จากนั้นจะมีข้อความ test ที่ debug



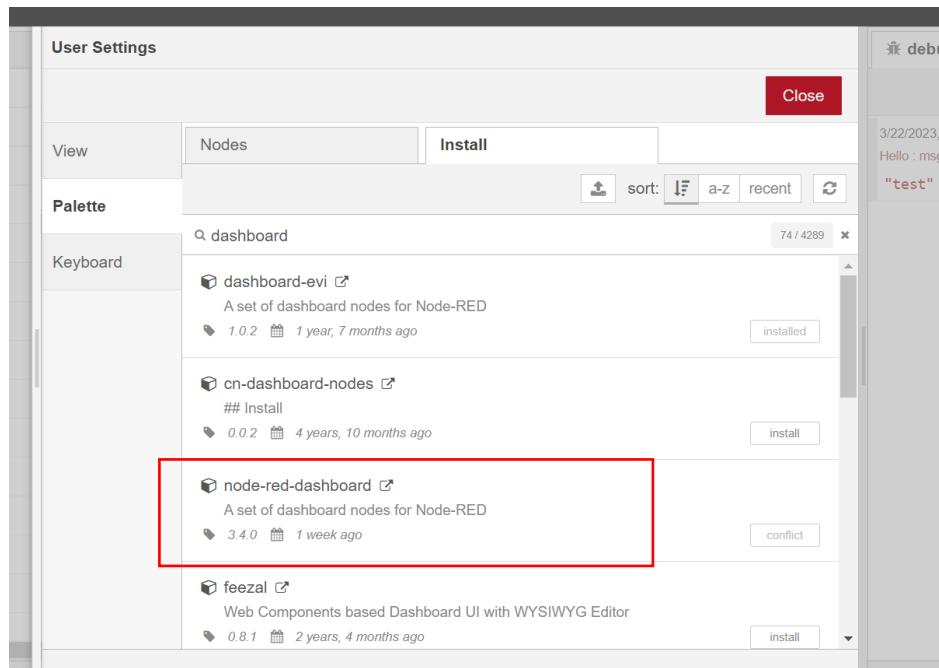
## 5.2 วิธีการ Publish ข้อมูลจาก MIC SMART ไปที่ Node-red

### ให้ทำการติดตั้ง palette node-red-contrib-aedes

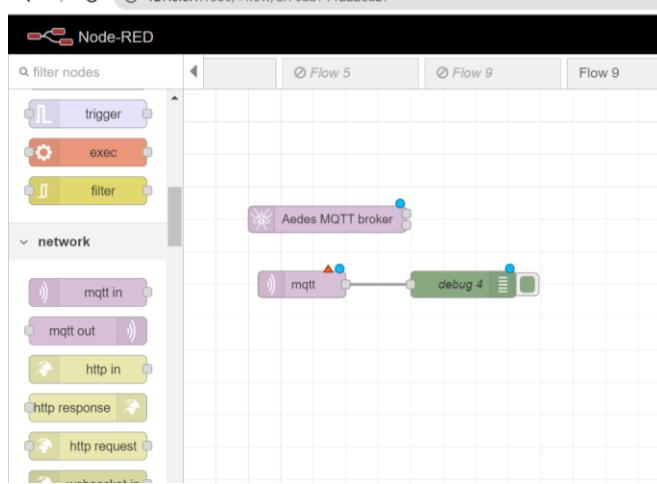
Download MQTT broker “Aedes”



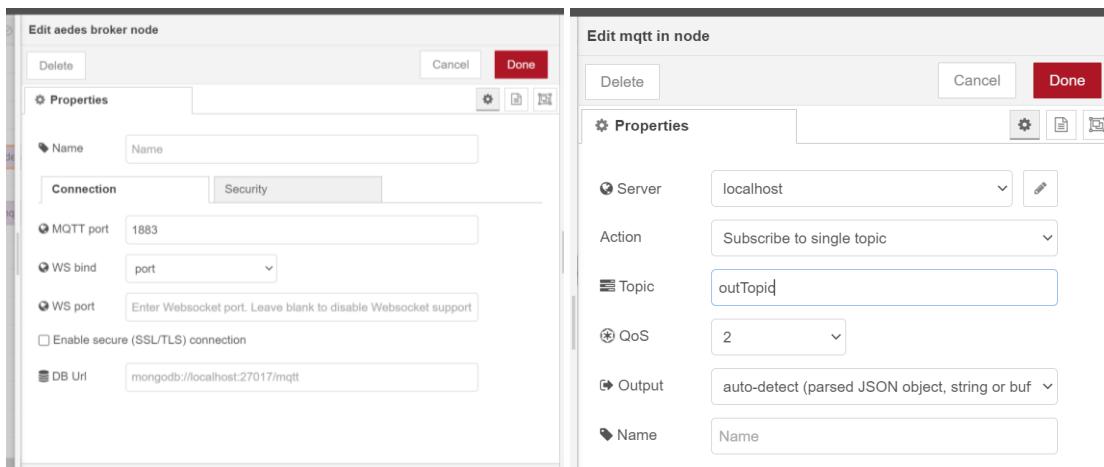
### ติดตั้ง palette node-red-dashboard



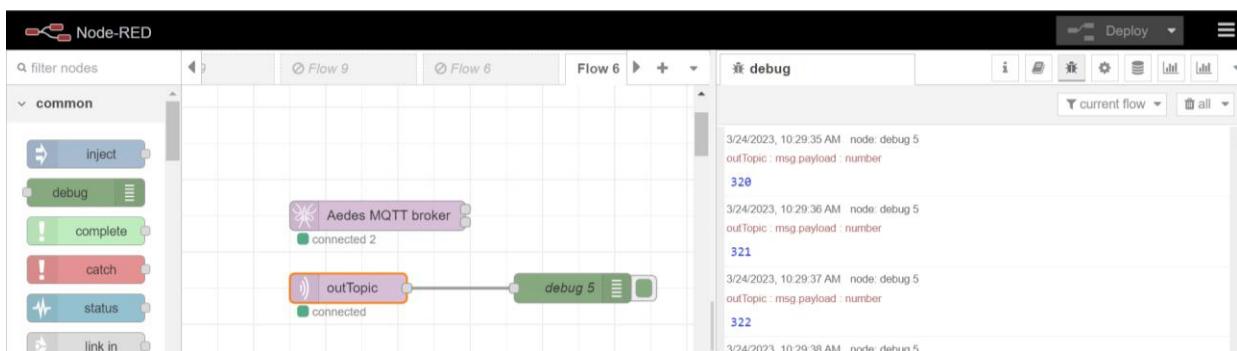
ทำการสร้าง Flow ลาก Aedes MQTT broker, mqtt in, debug วางใน Flow และเชื่อมต่อกัน ดังรูป



Edit addes broker node และ Edit mqtt in node ดังรูป



นำโค้ด Arduino IDE ในหัวข้อ 4.11 PubSubClient มาใช้งานเพื่อส่งค่าไป Topic ที่ชื่อว่า “outTopic” ผลลัพท์ที่ได้จะปรากฏ Data ขึ้นใน debug



## 5.3 วิธีการรับค่าจาก Modbus Slave ส่ง Data ขึ้น Node-red

ให้ทำการแก้ไขโค๊ด Arduino IDE โดยการรวมตัวอย่างในหัวข้อ 4.13 Modbus Master, 4.15 OTA Static IP, 4.16 PubSubClient เข้ามาไว้ในโค๊ดโปรแกรมเดียวกัน

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <ESPmDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>
#include <ModbusMaster.h>
```

```
ModbusMaster node;
```

```
const char* ssid = "TP-Link_2B32";
const char* password = "58252017";
const char* mqtt_server = "192.168.1.101"; //IP Server
```

```
//////////SETUP//////////
```

```
IPAddress local_IP(192, 168, 1, 208); // Static IP address
IPAddress gateway(192, 168, 1, 1); // Gateway IP address
IPAddress subnet(255, 255, 255, 0); // subnet
char machine[5] = "TB02"; //Machine name
//////////SETUP//////////
```

```
WiFiClient espClient;
```

```
PubSubClient client(espClient);
```

```
#define led_connection 5
```

```
#define led_run6
```

```
int count_connection;
```

```
void setup_wifi()
{
    delay(10);
    // We start by connecting to a WiFi network
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);
```

```

//WiFi.mode(WIFI_STA);
WiFi.config(local_IP, gateway, subnet);
WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {
    count_connection++;
    digitalWrite(led_connection, HIGH);delay(100);digitalWrite(led_connection, LOW);delay(100);
    Serial.print(".");
    if(count_connection>20)
    {
        ESP.restart();
    }
}

randomSeed(micros());
Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length)
{
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("] ");
    for (int i = 0; i < length; i++) {
        Serial.print((char)payload[i]);
    }
    Serial.println();
    // Switch on the LED if an 1 was received as first character
    if ((char)payload[0] == '1') {
        digitalWrite(BUILTIN_LED, LOW); // Turn the LED on (Note that LOW is the voltage level
        // but actually the LED is on; this is because
        // it is active low on the ESP-01)
    } else {
        digitalWrite(BUILTIN_LED, HIGH); // Turn the LED off by making the voltage HIGH
    }
}

```

```

int time_count;

void reconnect()
{
    // Loop until we're reconnected
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        // Create a random client ID
        String clientId = "ESP8266Client-";
        clientId += String(random(0xffff), HEX);
        // Attempt to connect
        if (client.connect(clientId.c_str())) {
            Serial.println("connected");
            // Once connected, publish an announcement...
            //client.publish("outTopic", "hello world");
            // ... and resubscribe
            client.subscribe("inTopic");
        } else {
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println(" try again in 5 seconds");
            // Wait 5 seconds before retrying
            //delay(5000);
            time_count++;
            digitalWrite(led_connection, HIGH);delay(500);digitalWrite(led_connection, LOW);delay(500);
            if(time_count>=10)
            {
                ESP.restart();
            }
        }
    }
}

void setup()
{
    pinMode(led_connection, OUTPUT); // Initialize the BUILTIN_LED pin as an output
    pinMode(led_published, OUTPUT); ///
    Serial.begin(9600);
    Serial1.begin(9600);
}

```

```

setup_wifi();

// Hostname defaults to esp3232-[MAC]
ArduinoOTA.setHostname(machine);

// No authentication by default
ArduinoOTA.setPassword("1234");

// Password can be set with it's md5 value as well
// MD5(admin) = 21232f297a57a5a743894a0e4a801fc3
// ArduinoOTA.setPasswordHash("21232f297a57a5a743894a0e4a801fc3");

ArduinoOTA

.onStart([]) {
    String type;
    if (ArduinoOTA.getCommand() == U_FLASH)
        type = "sketch";
    else // U_SPIFFS
        type = "filesystem";
    // NOTE: if updating SPIFFS this would be the place to unmount SPIFFS using SPIFFS.end()
    Serial.println("Start updating " + type);
}

.onEnd([]) {
    Serial.println("\nEnd");
}

.onProgress([](unsigned int progress, unsigned int total) {
    Serial.printf("Progress: %u%%\r", (progress / (total / 100)));
})

.onError([](ota_error_t error) {
    Serial.printf("Error[%u]: ", error);
    if (error == OTA_AUTH_ERROR) Serial.println("Auth Failed");
    else if (error == OTA_BEGIN_ERROR) Serial.println("Begin Failed");
    else if (error == OTA_CONNECT_ERROR) Serial.println("Connect Failed");
    else if (error == OTA_RECEIVE_ERROR) Serial.println("Receive Failed");
    else if (error == OTA_END_ERROR) Serial.println("End Failed");
});

ArduinoOTA.begin();

client.setServer(mqtt_server, 1883);
client.setCallback(callback);
node.begin(1, Serial1);
}

```

```

int counter;

String rss,d200,d201,d202,d203,d204,d205; // Add register ,d206,.....n;
char r[16],d0[16],d1[16],d2[16],d3[16],d4[16],d5[16]; // Add register ,d6[16],.....n;
int num = 1; //number of registers

void loop()
{
    ArduinoOTA.handle();
    uint8_t j, result;
    uint16_t data[num];
    if (!client.connected()) {
        reconnect();
    }
    client.loop();
    digitalWrite(led_connection, HIGH);
    Serial.println("\n-----starting loop-----");
    result = node.readHoldingRegisters(0, num);
    if (result == node.ku8MBSuccess)
    {
        for (j = 0; j < num; j++)
        {
            data[j] = node.getResponseBuffer(j);
        }
        d200 = String(data[0]);
        // Add register Ex. d201= String(data[1]);

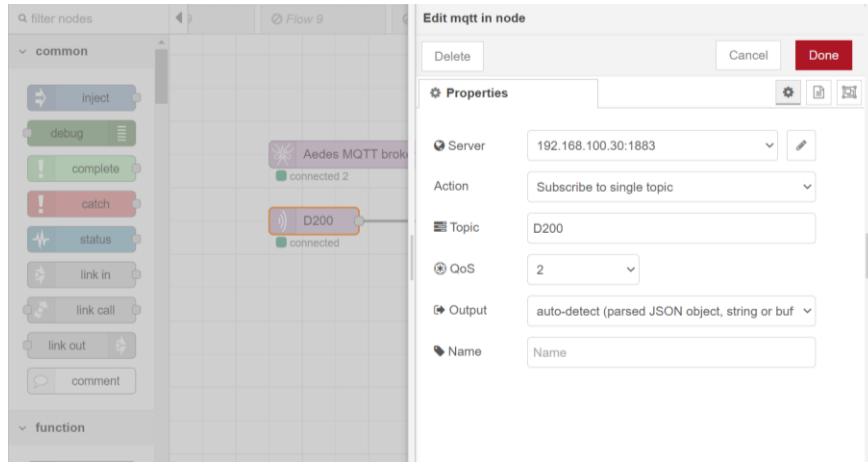
        digitalWrite(led_published, LOW);delay(500);digitalWrite(led_published, HIGH);delay(500); //////
    }
    rss = WiFi.RSSI(); // WiFi strength
    Serial.print("rss : ");Serial.println(WiFi.RSSI());
    Serial.print("d200 : ");Serial.println(d200);
    // Add Serial Monitor

    d200.toCharArray(d0,16);client.publish("TB01/D200", d0);
    // Add register Ex. d201.toCharArray(d1,16);client.publish("D201", d1);

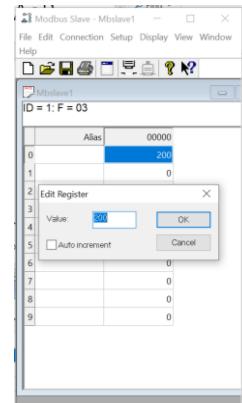
    Serial.println("\n-----finish loop-----\n\n");
    delay(5000);
}

```

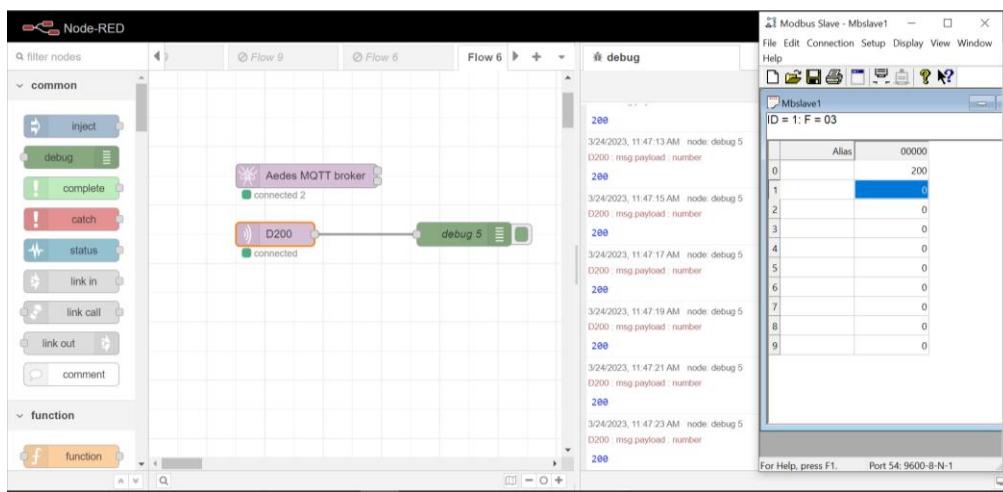
Node red Edit mqtt in node ไปยัง Topic เป็น D200



Modbus Slave Edit register value : 200 จากนั้น กด OK

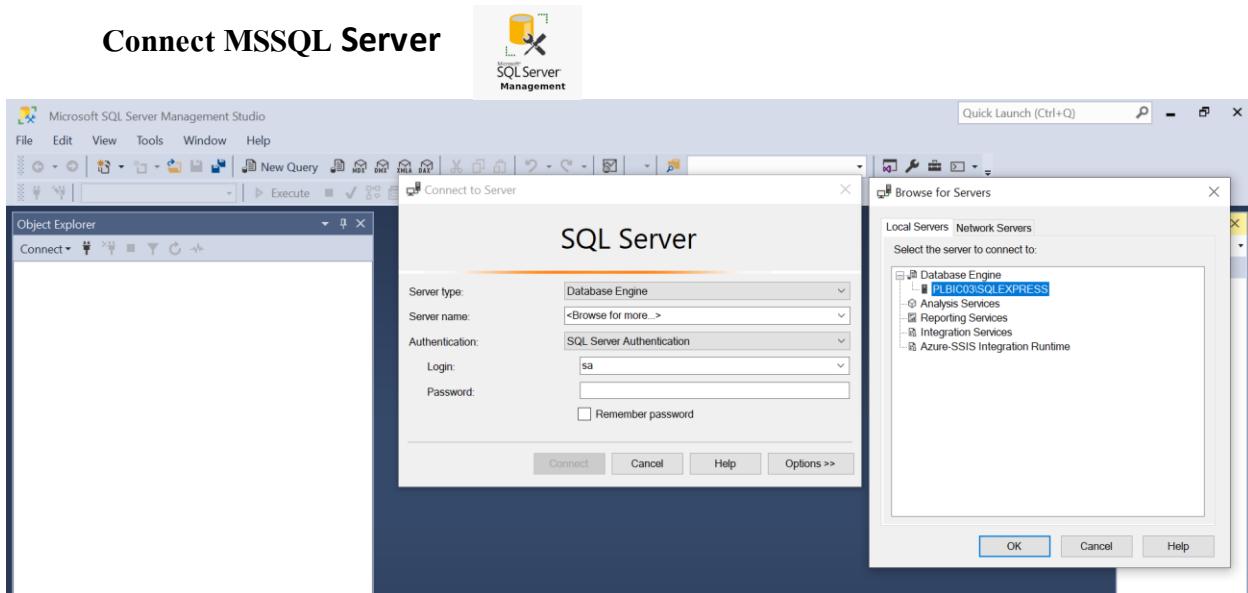


เมื่อ MIC-SMART รับค่า Address 0 : 200 จาก Modbus Slave และจะ publish ขึ้น Node-red ด้วย Topic ที่ชื่อว่า D200



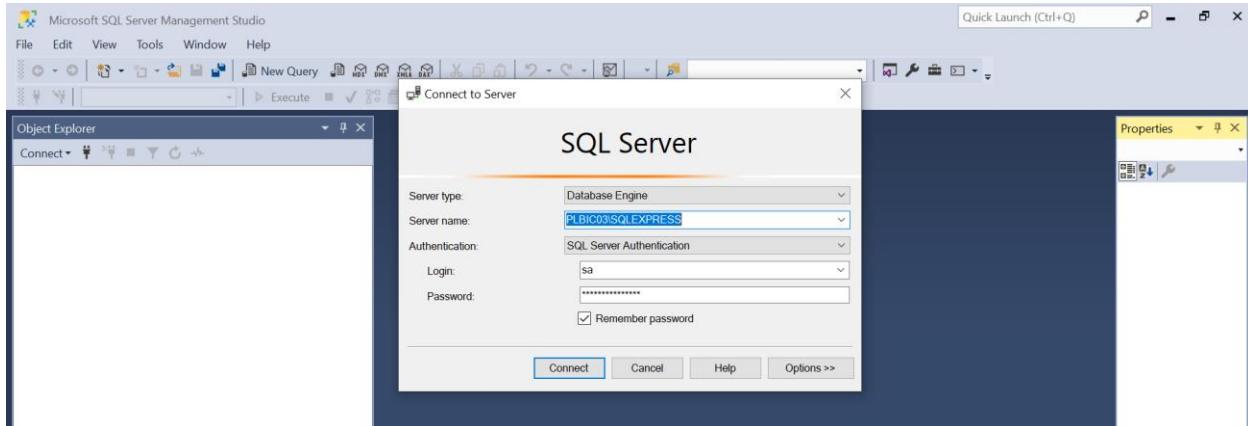
## 5.4 วิธีการนำค่าที่ได้เข้าไปใน SQL Server

นำโค๊ด Arduino ในหัวข้อ 5.3 upload to MIC-SMART เพื่อส่ง data ขึ้น Node-red

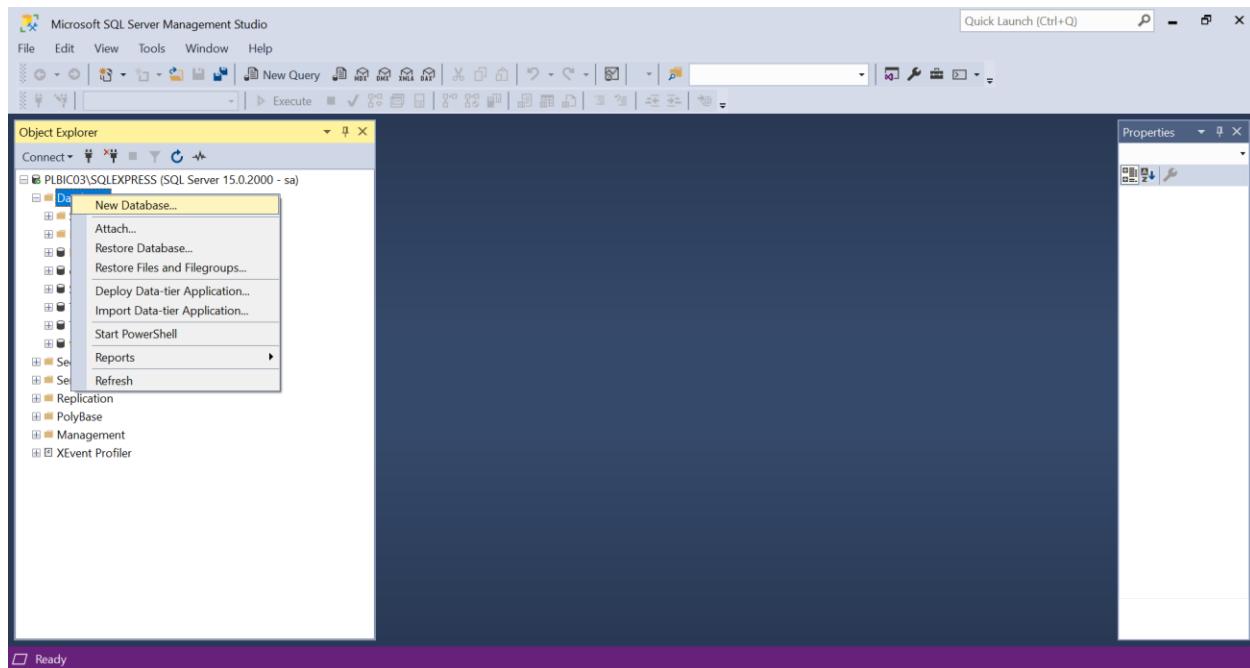


Login : sa

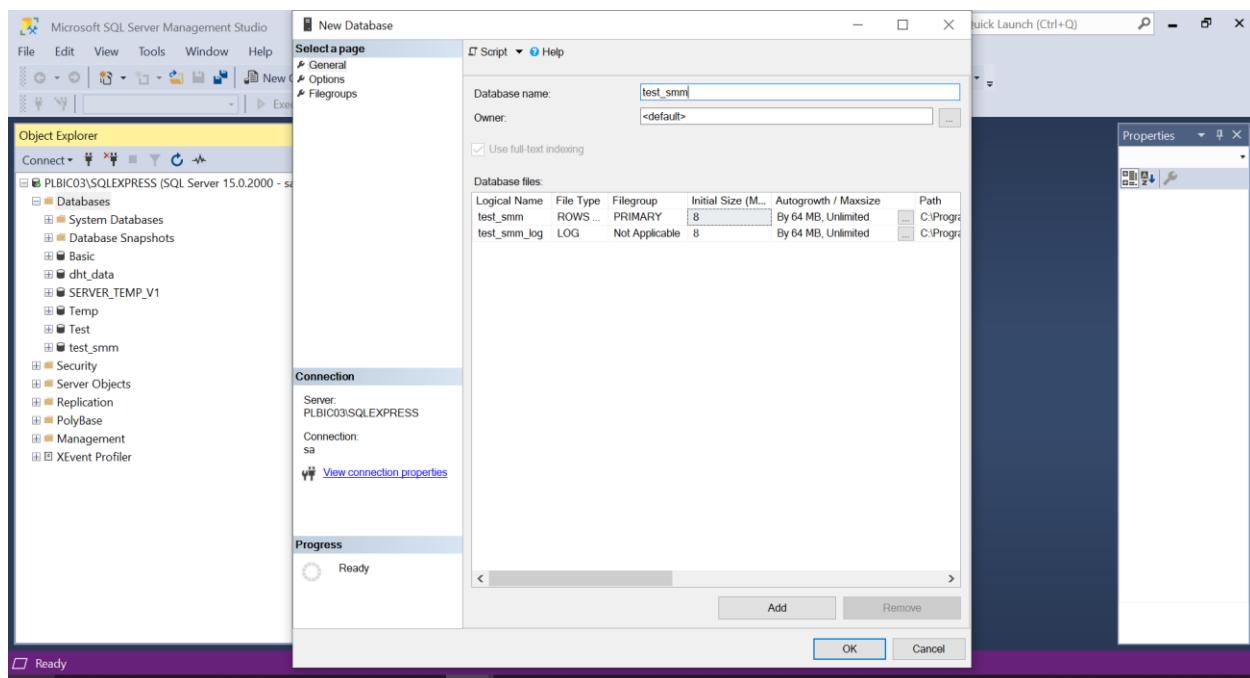
Password: sa@admin



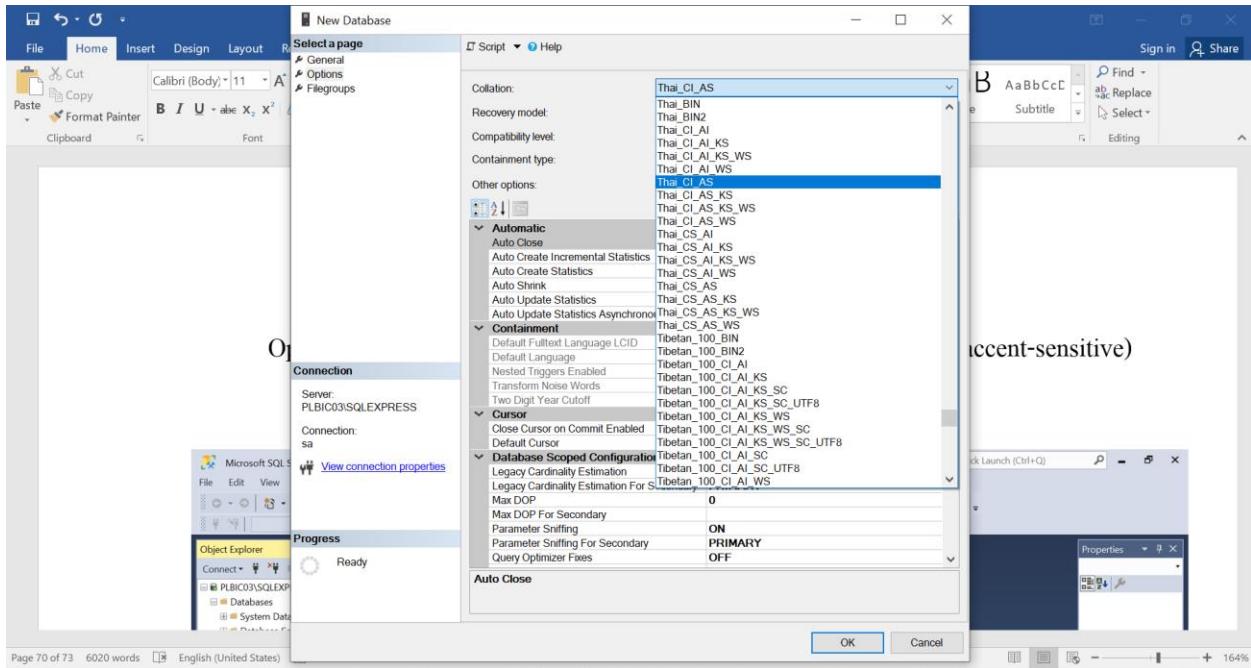
คลิกขวาที่ Databases เลือก New Database....



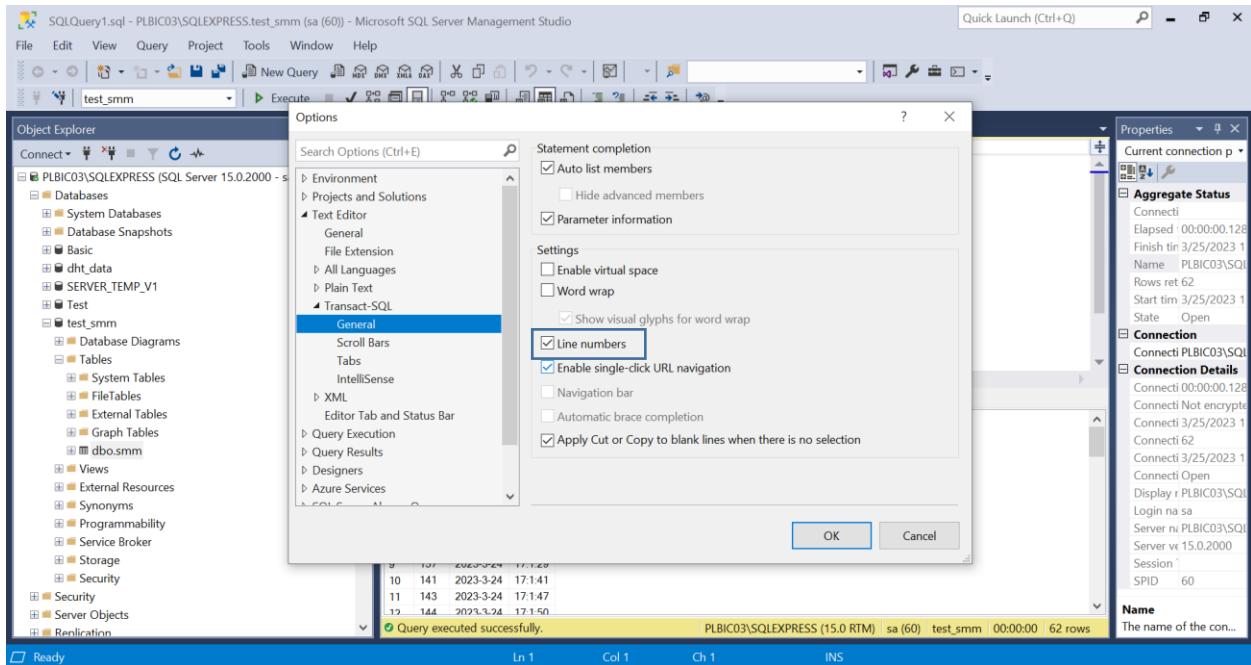
General ตั้งชื่อ Database name: test\_smm



Options เลือก Collation เป็น Thai\_CI\_AS (Thai , case-insensitive, accent-sensitive)

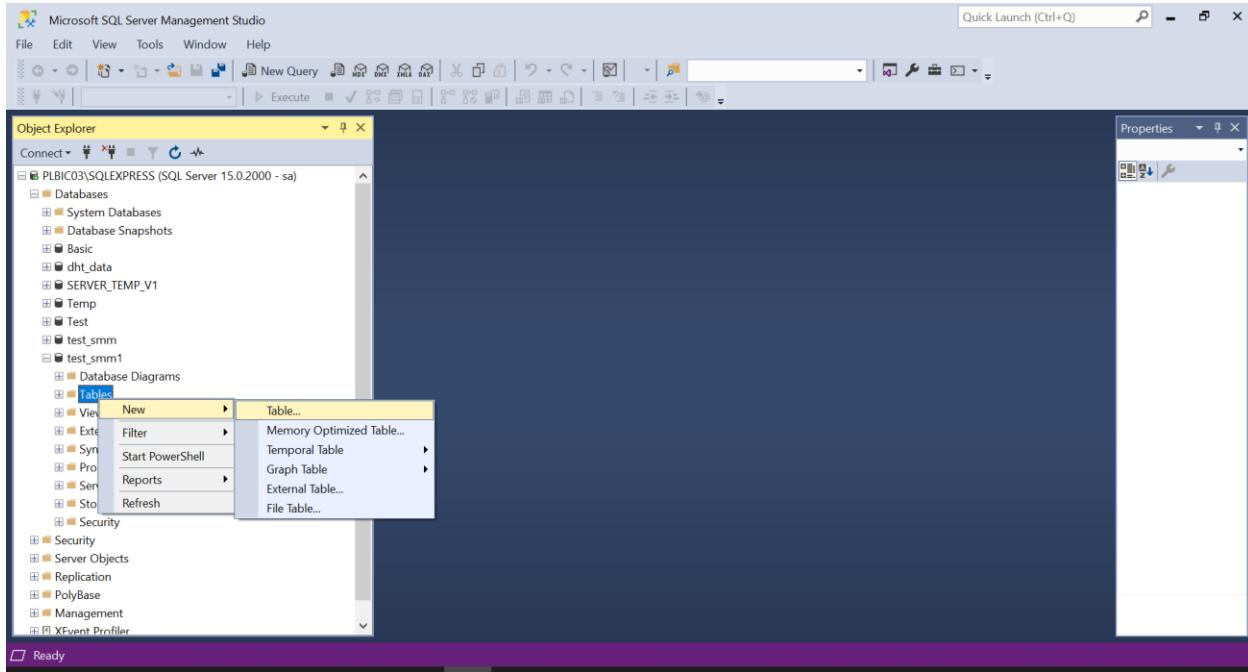


Tools → Options →Text Editor →General→ Check the Line numbers option



## Create Table

Tables → New → Table...



สร้างข้อมูล Register ,Value, date, time ดังรูป

Column Name	Data Type	Allow Nulls
Register	nchar(10)	<input checked="" type="checkbox"/>
Value	nchar(10)	<input checked="" type="checkbox"/>
date	date	<input checked="" type="checkbox"/>
time	time(7)	<input checked="" type="checkbox"/>

ใน SQL มีชนิดของข้อมูล (Data Types) ที่ใช้เก็บข้อมูลได้หลากหลาย แต่ละชนิดของข้อมูล จะมีการใช้งานและข้อจำกัดที่แตกต่างกันไปตามลักษณะของข้อมูล ดังนี้

### ข้อมูลชนิดตัวเลข (Numeric Data Types)

INT: เก็บค่าเลขจำนวนเต็ม (integer)

BIGINT: เก็บค่าเลขจำนวนเต็มขนาดใหญ่

SMALLINT: เก็บค่าเลขจำนวนเต็มขนาดเล็ก

TINYINT: เก็บค่าเลขจำนวนเต็มขนาดเล็กที่สุด

FLOAT: เก็บค่าทศนิยม (floating-point number) แบบเลขจำนวนจริง (real number)

DOUBLE: เก็บค่าทศนิยมแบบเลขจำนวนจริงขนาดใหญ่

### ข้อมูลชนิดข้อความ (Character Data Types)

CHAR: เก็บข้อความคงที่ (fixed-length string)

VARCHAR: เก็บข้อความตัวแปร (variable-length string)

TEXT: เก็บข้อความยาว (long string)

### ข้อมูลชนิดวัน-เวลา (Date and Time Data Types)

DATE: เก็บวันที่ (date)

TIME: เก็บเวลา (time)

DATETIME: เก็บวันที่และเวลา

TIMESTAMP: เก็บเวลาสำหรับเป็นค่าความสมบูรณ์ของเวลา (absolute time)

### ข้อมูลชนิดตรรกะ (Boolean Data Type)

BOOLEAN: เก็บค่าจริง (true) หรือเท็จ (false)

ข้อมูลชนิดอื่นๆ

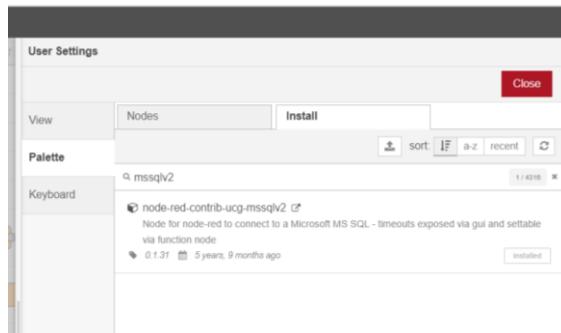
BLOB: เก็บข้อมูลแบบไบนารีใหญ่ (binary large object)

JSON: เก็บข้อมูลในรูปแบบ JSON (JavaScript Object Notation)

XML: เก็บข้อมูลในรูปแบบ XML (Extensible Markup Language)

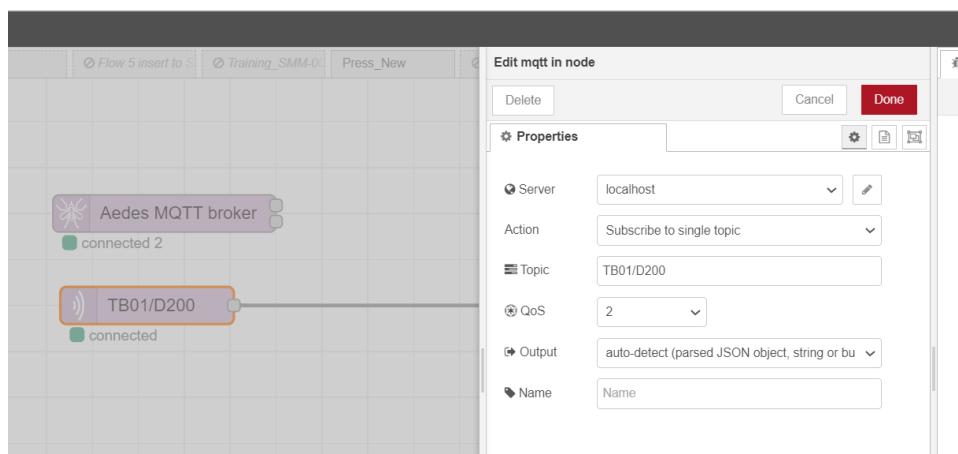
## สร้าง Flow Node-red เพื่อทำการ Insert Data

ให้ทำการติดตั้ง palette node-red-contrib-ucg-mssqlv2



Edit mqtt in node

Topic : TB01/D200

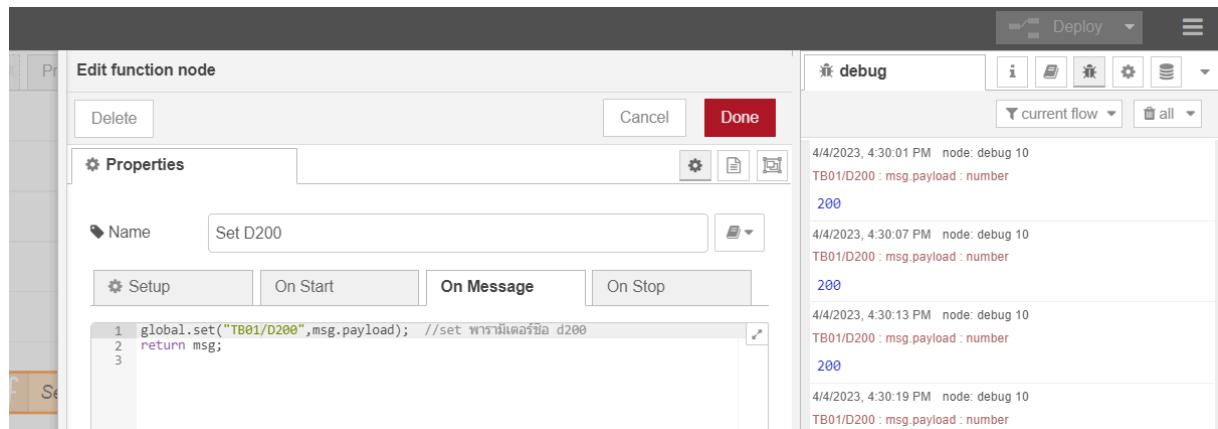


เพิ่ม Function วางที่ Flow แล้วเชื่อมต่อดังรูป ตั้งชื่อ Set D200 ใน On Message ใส่โค้ดดังนี้

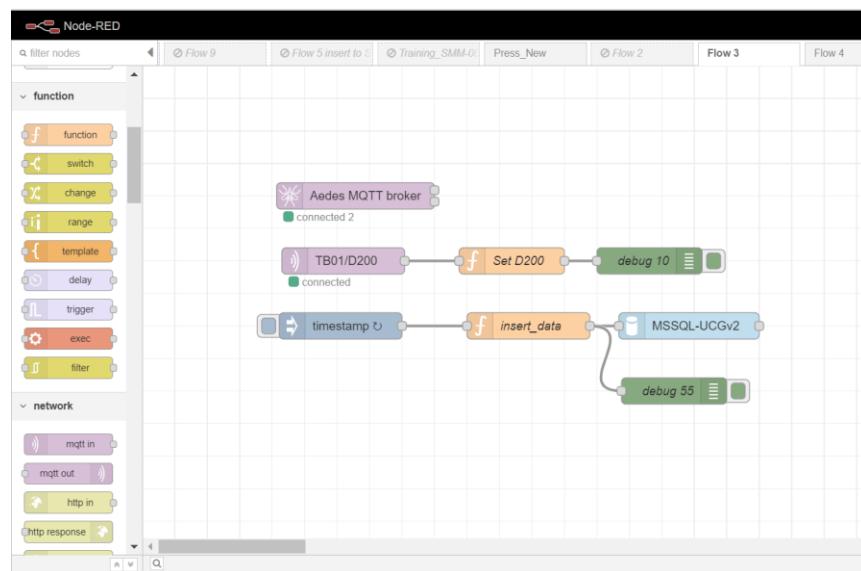
```
global.set("TB01/D200",msg.payload); //set พารามิเตอร์ชื่อ TB01/D200
```

```
return msg;
```

จากนั้นกด Debug แล้ว กด Deploy ผลลัพท์ที่ได้ดังรูป



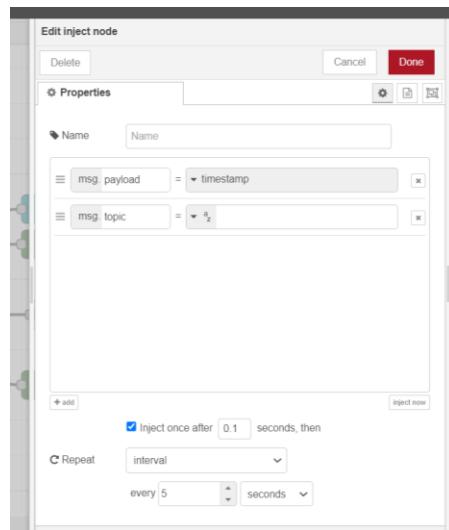
เพิ่ม inject, Function, MSSQL-UCGv2, debug เข้ามต่อดังรูป



### Edit inject node

Interval

Every 5 seconds



### Edit function node

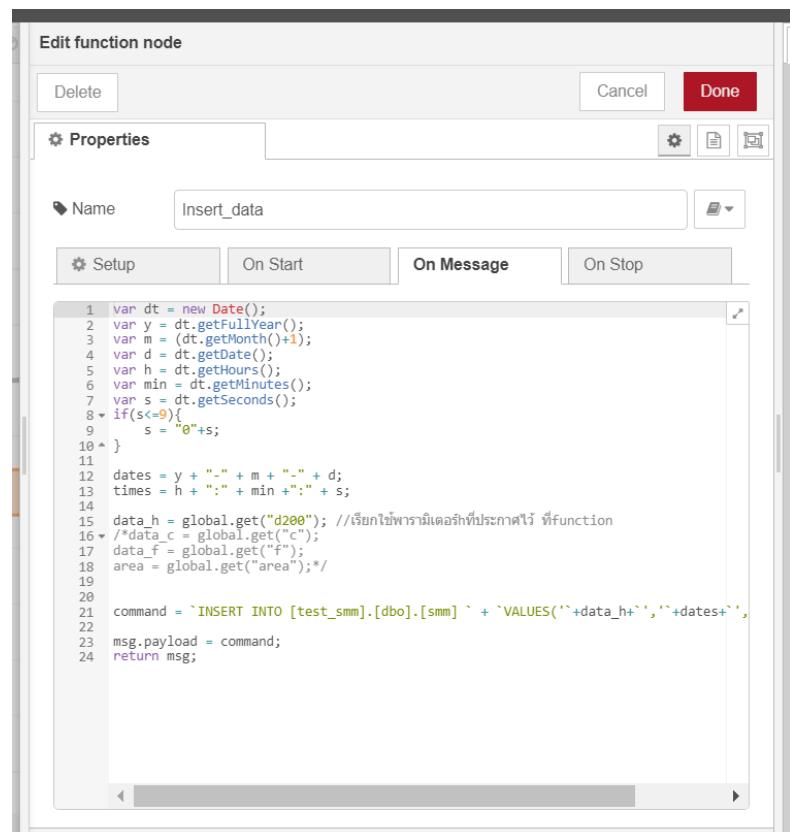
Name insert\_data

เพิ่มโค้ดใน On Message

```

var dt = new Date();
var y = dt.getFullYear();
var m = (dt.getMonth()+1);
var d = dt.getDate();
var h = dt.getHours();
var min = dt.getMinutes();
var s = dt.getSeconds();
if(s<=9){
  s = "0"+s;
}
var dates = y + "-" + m + "-" + d;
var times = h + ":" + min + ":" + s;

```



```

var data_200 = global.get("TB01/D200");
var pin = "TB01/D200";

var command = `INSERT INTO [test_smm].[dbo].[smm] ` +
'VALUES(`'+pin+'`,'+data_200+','+dates+','+times+')';
msg.payload = command;

return msg;

```

Edit MSSQL-UCGv2 node > Edit MSSQL-UCGv2-CN node

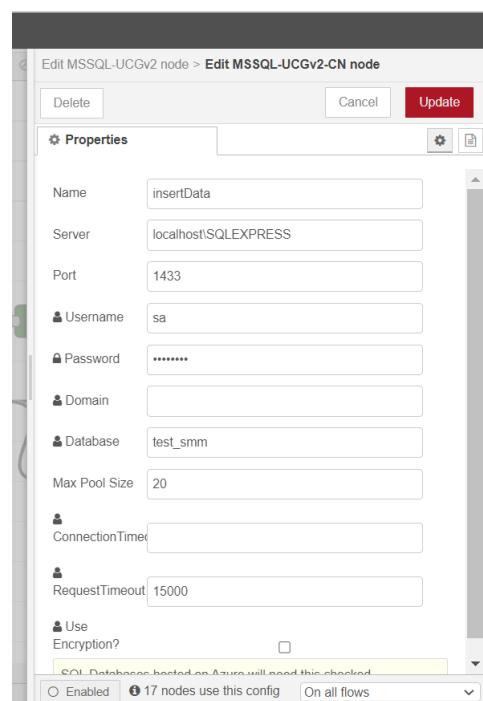
Name InsertData

Server localhost\SQLEXPRESS

Username sa

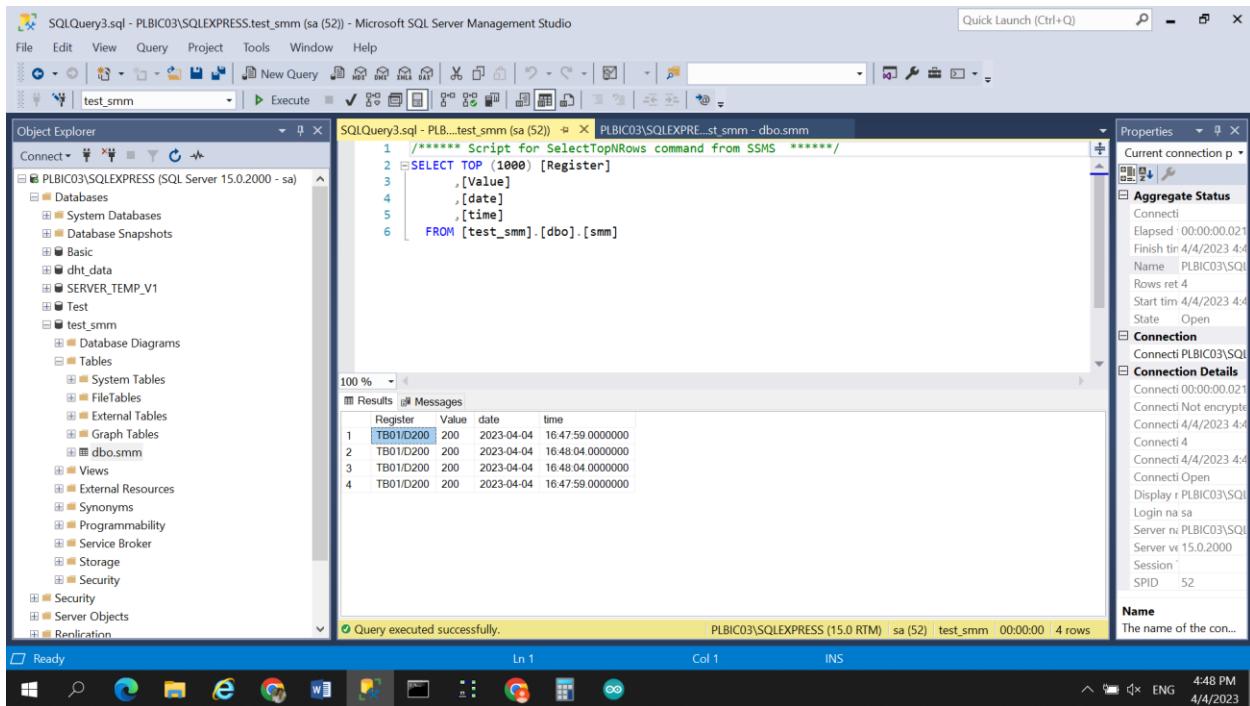
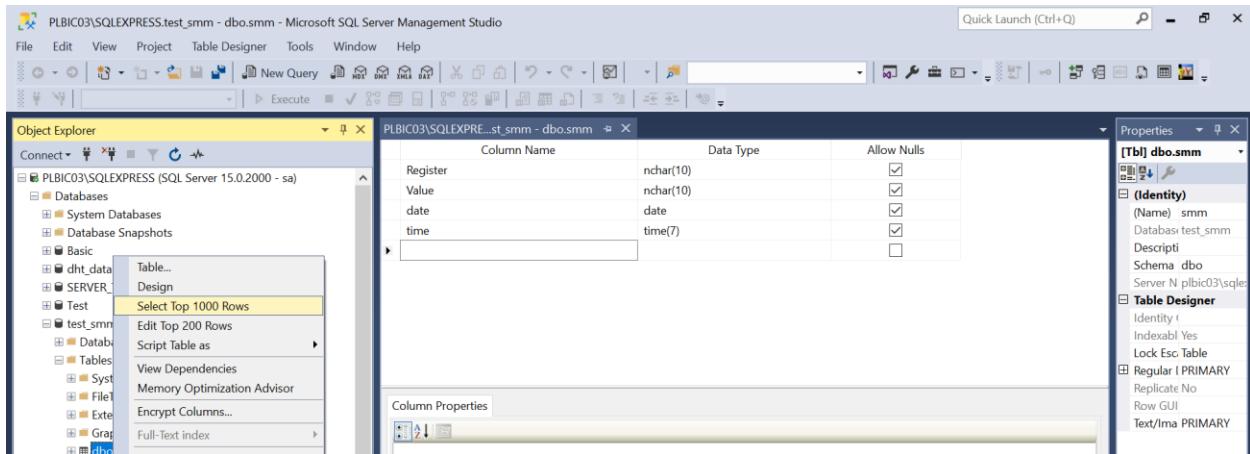
Password sa@admin

Database test\_smm



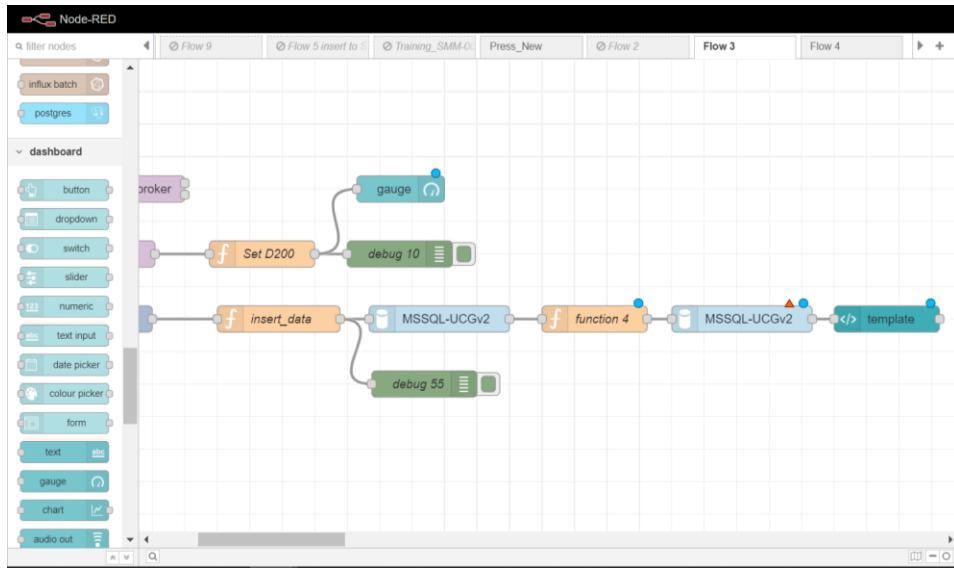
กด Update และ Deploy จากนั้น Data ที่ได้จะบันทึกเข้า SQL

## วิธีดู Data ให้คลิกขวาที่ dbo.smm เลือก Select Top 1000 Rows



## สร้าง Flow Node-red เพื่อกำกານ Select Data

เพิ่ม Function, MSSQL-UCGv2, template, gauge เชื่อมต่อดังรูป



### Edit function node

Name top50

เพิ่มโค้ดใน On Message

```
var sqlCommand = ('SELECT TOP (50) [date]
```

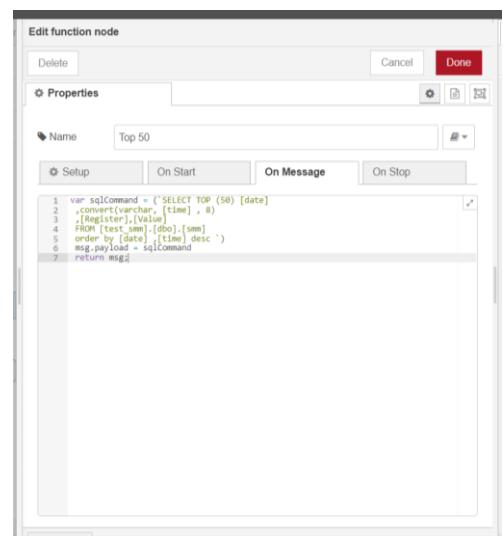
```
,convert(varchar, [time] , 8),[Register],[Value]
```

```
FROM [test_smm].[dbo].[smm]
```

```
order by [date] ,[time] desc `)
```

```
msg.payload = sqlCommand
```

```
return msg;
```



Edit MSSQL-UCGv2 node > Edit MSSQL-UCGv2-CN node

Name getTable

Server localhost\SQLEXPRESS

Username sa

Password sa@admin

Database test\_smm

Edit template node > Edit dashboard group node > Edit dashboard tab node

Name : Monitering

Edit template node > Edit dashboard group node

Name : Page\_1

Tab : Monitering

Edit template node

Group: [Monitering Page\_1]

Template ເພີ່ມໂຄດ

```
<table id = "table" border ="1" >
  <tr>
```

```
    <th> Date </th>
```

```
    <th> time </th>
```

```
    <th> Register </th>
```

```
    <th> Value </th>
```

```
  </tr>
```

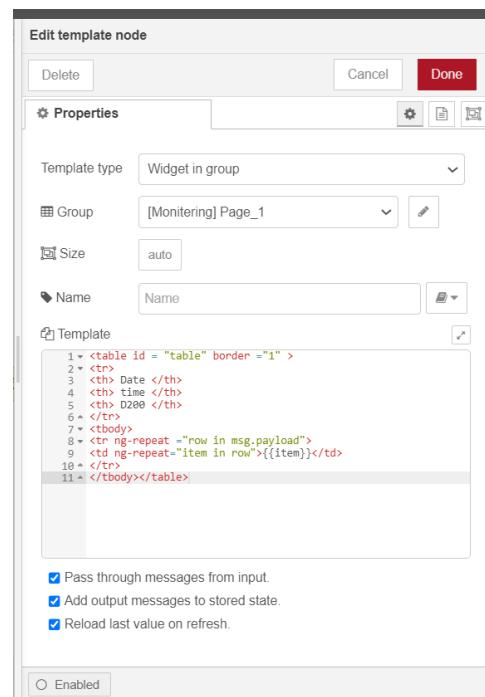
```
<tbody>
```

```
  <tr ng-repeat ="row in msg.payload">
```

```
    <td ng-repeat="item in row">{ {item} }</td>
```

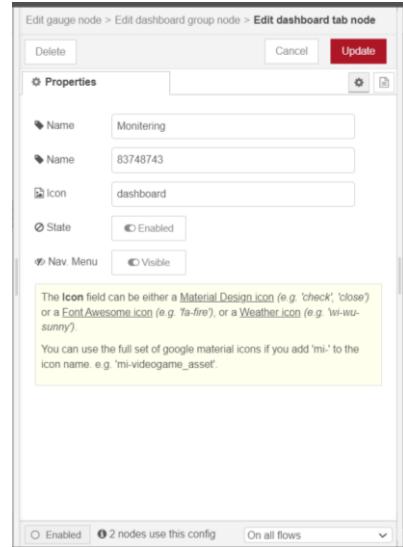
```
  </tr>
```

```
</tbody></table>
```



Edit gauge node > Edit dashboard group node > Edit dashboard tap node

Name Monitoring

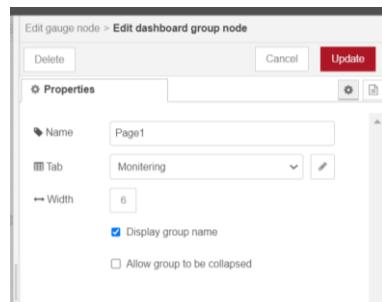


Edit gauge node > Edit dashboard group node

Name Page\_1

Tab Monitering

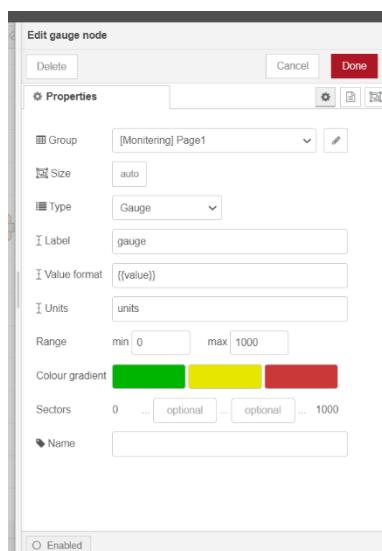
Width 6



Edit gauge node

Group [Monitering] Page\_1

Range min 0 max 1000



สามารถดู Dashboard ได้ที่ <http://127.0.0.1:1880/ui>

