สารบัญ: Senses-Hub AI IoT

จากเซ็นเซอร์สู่ผู้ช่วย AI อัจฉริยะด้วย n8n

บทที่ 1: เริ่มต้นกับฮาร์ดแวร์ (Getting Started with Hardware)

บทที่ 2: การเขียนโปรแกรมควบคุมบอร์ดและเซ็นเซอร์ (Programming the Board)

บทที่ 3: รู้จักกับ n8n แพลตฟอร์มเชื่อมทุกอย่างเข้าด้วยกัน (Introduction to n8n)

- **3.1 n8n คืออะไร?**
  - แนวคิด Workflow Automation และ Low-code/No-code
- **3.2 การติดตั้งและเริ่มต้นใช้งาน n8n**
  - **การติดตั้งแบบ Local บนคอมพิวเตอร์ (แนะนำ):**
    - การติดตั้งผ่าน Docker
  - **ทางเลือกเสริม:** การใช้งานผ่าน n8n Cloud
- **3.3 ส่วนประกอบหลักของ n8n ที่ต้องรู้**
  - **Workflow:** ผืนผ้าใบสำหรับการสร้างระบบอัตโนมัติ
  - **Node:** หน่วยการทำงาน (เช่น Trigger, Action)
  - **Connection:** เส้นเชื่อมการทำงานระหว่าง Node
  - **Credential:** การจัดการข้อมูลการยืนยันตัวตน (API Keys, Login)
- **3.4 สร้าง Workflow แรก: "Hello, n8n!"**
  - ทดลองใช้ **Manual Trigger Node** และ **Set Node** เพื่อทำความเข้าใจการไหลของข้อมูล

บทที่ 4: การเชื่อมต่อบอร์ด IoT กับ n8n (Connecting IoT to n8n)

- **4.1 การส่งข้อมูลจาก ESP32 สู่ n8n**
  - **วิธีที่ 1: Webhook (ง่ายและรวดเร็ว)**
    - การสร้าง **Webhook Node** ใน n8n เพื่อรอรับข้อมูล
    - เขียนโค้ด ESP32 ให้เชื่อมต่อ Wi-Fi และส่งข้อมูลเซ็นเซอร์ผ่าน HTTP POST Request
  - **วิธีที่ 2: MQTT (มาตรฐานสำหรับ IoT)**
    - ภาพรวมหลักการทำงานของ MQTT (Publisher, Subscriber, Broker)
    - การใช้ **MQTT Trigger Node** ใน n8n
    - เขียนโค้ด ESP32 เพื่อส่งข้อมูลไปยัง MQTT Broker
- **4.2 การจัดการและแปลงข้อมูลใน n8n**
  - ใช้ **Set Node** และ **Function Node (JavaScript)** เพื่อจัดรูปแบบข้อมูล JSON ที่ได้รับจาก IoT ให้อยู่ในรูปแบบที่พร้อมใช้งาน

บทที่ 5: การจัดเก็บและจัดการข้อมูลด้วย PostgreSQL (Data Persistence with PostgreSQL)

- **5.1 ทำไมต้องมีฐานข้อมูล?**
- **5.2 การติดตั้ง PostgreSQL เบื้องต้น**
  - การติดตั้งผ่าน Docker หรือสมัครใช้บริการฐานข้อมูลบน Cloud
- **5.3 การเชื่อมต่อ n8n กับ PostgreSQL**
  - การตั้งค่า Credential สำหรับ PostgreSQL ใน n8n
  - การใช้งาน **Postgres Node**
- **5.4 Workflow: บันทึกข้อมูลเซ็นเซอร์ลงฐานข้อมูล**
  - ออกแบบ Workflow ที่รับข้อมูลจาก Webhook/MQTT แล้วนำไปบันทึกลงตารางในฐานข้อมูลโดยอัตโนมัติ

บทที่ 6: สร้างผู้ช่วยอัจฉริยะ (AI Agent) ด้วย n8n

- **6.1 หลักการทำงานของ AI Agent และ LLMs**
- **6.2 การเชื่อมต่อ Large Language Models (LLM)**
  - **6.2.1 การใช้ Cloud-based LLM (เช่น OpenAI, Google Gemini)**
    - การขอ API Key จากผู้ให้บริการ
    - การตั้งค่า Credential ใน n8n ผ่าน **LLM Node**
  - **6.2.2 การใช้ Local LLM (ทำงานบนเครื่องของคุณเพื่อความเป็นส่วนตัวและประหยัดค่าใช้จ่าย)**
    - **Ollama:** การติดตั้ง, การดาวน์โหลดโมเดล และการเรียกใช้งาน
    - **LM Studio:** การตั้งค่า Server และการนำโมเดลมาใช้
    - การเชื่อมต่อ n8n กับ Local LLM ผ่าน **Ollama Node** หรือ **HTTP Request Node**
- **6.3 ตัวอย่าง Workflow สำหรับ AI Agent**
  - **AI Agent #1: นักวิเคราะห์ข้อมูลสภาพอากาศ**
    - สร้าง Workflow ที่ทำงานตามเวลา (Schedule Trigger)
    - ดึงข้อมูลจาก PostgreSQL
    - ส่งข้อมูลให้ AI ช่วยสรุปและวิเคราะห์แนวโน้ม (เช่น "สรุปสภาพอากาศและแนวโน้มการเคลื่อนไหวใน 24 ชั่วโมงที่ผ่านมา")
    - ส่งผลการวิเคราะห์ไปที่ LINE หรือ Email
  - **AI Agent #2: ผู้ช่วยตอบคำถามจากข้อมูล Real-time**
    - สร้าง Workflow ที่รับคำถามผ่าน Webhook
    - ค้นหาข้อมูลล่าสุดจาก PostgreSQL
    - สร้าง Prompt เพื่อสั่งให้ AI ตอบคำถามโดยอิงจากข้อมูลจริง (เช่น "ตอนนี้อุณหภูมิเท่าไหร่?", "มีการสั่นสะเทือนผิดปกติหรือไม่?")
- **6.4 Node สำคัญอื่นๆ เพื่อสร้าง Workflow อัจฉริยะ**
  - **Switch Node:** สร้างเงื่อนไขเพื่อตัดสินใจ (เช่น ถ้าอุณหภูมิ > 35°C ให้ส่งการแจ้งเตือน)
  - **Merge Node:** รวมข้อมูลจากหลายแหล่งก่อนประมวลผล
  - **HTTP Request Node:** เชื่อมต่อกับ API ภายนอกอื่นๆ

ภาคผนวก

- **A: Pinout และข้อมูลทางเทคนิคของบอร์ด SH-BC3**
- **B: การแก้ไขปัญหาที่พบบ่อย (Troubleshooting)**
- **C: แหล่งข้อมูลและไลบรารี่เพิ่มเติม**

บทที่ 1: เริ่มต้นกับฮาร์ดแวร์ (Getting Started with Hardware)

ยินดีต้อนรับสู่คู่มือการสร้างโปรเจกต์ IoT และ AI ด้วยบอร์ด Senses-Hub SH-BC3! ในบทแรกนี้เราจะเริ่มต้นด้วยการทำความรู้จักกับฮาร์ดแวร์ซึ่งเป็นหัวใจสำคัญของโปรเจกต์, ภาพรวมสถาปัตยกรรมทั้งหมด, และการเตรียมความพร้อมเครื่องมือที่จำเป็น เพื่อให้คุณมั่นใจว่าพร้อมสำหรับการเดินทางสู่โลกแห่งระบบอัตโนมัติและปัญญาประดิษฐ์

1.1 แนะนำบอร์ด Senses-Hub

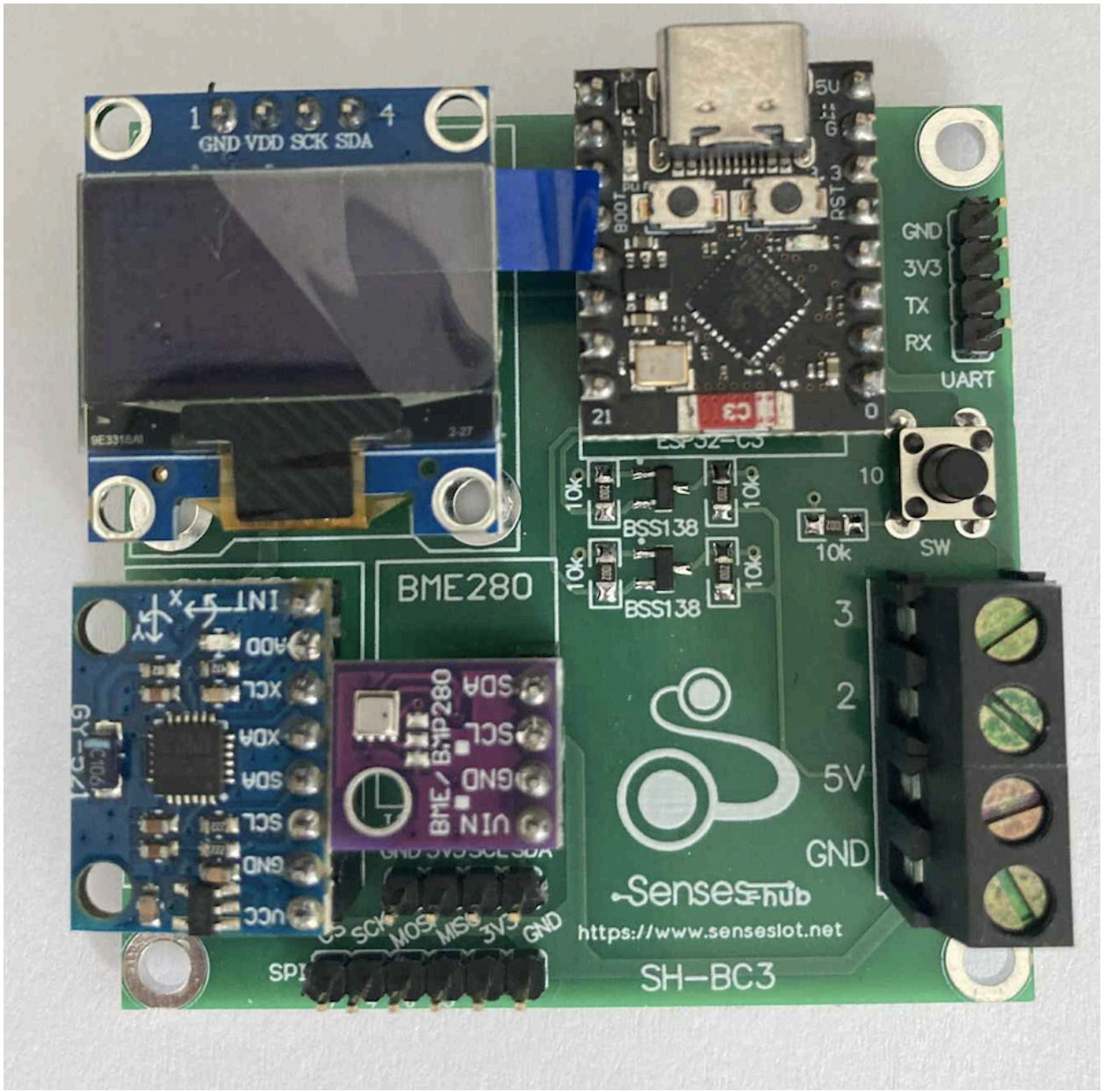
Senses-Hub SH-BC3 คือบอร์ดพัฒนา (Development Board) ที่ถูกออกแบบมาเพื่อเป็นจุดเริ่มต้นที่สมบูรณ์แบบสำหรับโปรเจกต์ IoT (Internet of Things) โดยรวบรวมส่วนประกอบที่จำเป็นไว้ในบอร์ดเดียว ทำให้คุณสามารถสร้างโปรเจกต์ที่ซับซ้อนได้โดยไม่ต้องต่อวงจรเพิ่มเติมให้ยุ่งยาก

ภาพรวมคุณสมบัติและความสามารถ:

- **All-in-One: รวม Microcontroller, เซ็นเซอร์หลากหลาย, และจอแสดงผลไว้บนบอร์ดเดียว**
- **Compact Size: ขนาดกะทัดรัด เหมาะสำหรับติดตั้งในพื้นที่จำกัด**
- **Easy to Use: มีจุดเชื่อมต่อมาตรฐาน ทำให้ง่ายต่อการขยายการทำงานในอนาคต**
- **Low Power: ใช้ชิป ESP32-C3 ที่ประหยัดพลังงาน เหมาะสำหรับโปรเจกต์ที่ต้องทำงานต่อเนื่อง**

ทำความรู้จักส่วนประกอบหลักบนบอร์ด:

- **Microcontroller (MCU): ESP32-C3 คือสมองกลของบอร์ด มาพร้อมความสามารถในการเชื่อมต่อ Wi-Fi และ Bluetooth Low Energy (BLE) ในตัว**
- **Sensors:**
    - **BME280: เซ็นเซอร์วัดสภาพแวดล้อม สามารถวัด อุณหภูมิ, ความชื้น, และความกดอากาศ ได้อย่างแม่นยำ**
    - **GY-521 (MPU-6050): เซ็นเซอร์ตรวจจับการเคลื่อนไหวแบบ 6 แกน (6-Axis) ประกอบด้วย Accelerometer (วัดความเร่ง 3 แกน) และ Gyroscope (วัดการหมุน 3 แกน) เหมาะสำหรับตรวจจับการสั่นสะเทือน, การเอียง, หรือการเคลื่อนที่**
- **Display: จอแสดงผล OLED ขนาดเล็กสำหรับแสดงข้อมูลสำคัญ เช่น ค่าจากเซ็นเซอร์ หรือสถานะการทำงาน โดยไม่ต้องเชื่อมต่อกับคอมพิวเตอร์ตลอดเวลา**
- **Connectivity:**
    - **I2C, UART, SPI: พอร์ตมาตรฐานสำหรับเชื่อมต่อกับเซ็นเซอร์หรืออุปกรณ์เสริมอื่นๆ**
    - **ช่องจ่ายไฟ 5V และ GND: สำหรับจ่ายไฟให้กับอุปกรณ์ภายนอก**


1 4
GND VDD SCK SDA
BME280
BSS138
BSS138
10k
10k
10k
10k
10k
10
SW
GND
3V3
TX
RX
UART
3
2
5V
GND
SDA
SCL
GND
VIN
BMP280
INT
ADO
XCL
XDA
SDA
SCL
GND
VCC
CS SCK MOSI MISO 3V3 GND
SPI
Sensesiot hub
https://www.senseslot.net
SH-BC3

1.2 สถาปัตยกรรมของโปรเจกต์

เพื่อให้เห็นภาพรวมทั้งหมดของโปรเจกต์ที่เรากำลังจะสร้าง ลองดูแผนภาพการทำงานง่ายๆ ด้านล่างนี้ ซึ่งแสดงเส้นทางการไหลของข้อมูลตั้งแต่ต้นทางไปจนถึงปลายทาง

Hardware (IoT) ⟶ n8n (Automation) ⟶ PostgreSQL (Database) ⟶ AI Agent (Intelligence)

1. **Hardware (Senses-Hub): ทำหน้าที่รวบรวมข้อมูลจากโลกจริง (อุณหภูมิ, การสั่นสะเทือน) แล้วส่งข้อมูลผ่าน Wi-Fi**
2. **n8n (Automation): ทำหน้าที่เป็น "บุรุษไปรษณีย์อัจฉริยะ" คอยรับข้อมูลจากบอร์ด, จัดการข้อมูล, และส่งต่อไปยังส่วนต่างๆ**
3. **PostgreSQL (Database): ทำหน้าที่เป็น "คลังข้อมูล" สำหรับเก็บข้อมูลที่ได้รับมาอย่างเป็นระบบ เพื่อนำไปใช้งานต่อในอนาคต**
4. **AI Agent (Intelligence): ทำหน้าที่เป็น "นักวิเคราะห์" โดยจะดึงข้อมูลจากคลังข้อมูลมาวิเคราะห์, สรุปผล, หรือตอบคำถามตามที่เราต้องการผ่าน Local LLM**

1.3 การเตรียมความพร้อมก่อนเริ่ม

ก่อนที่เราจะเริ่มเขียนโค้ด เราต้องติดตั้งโปรแกรมและเครื่องมือที่จำเป็นบนคอมพิวเตอร์ของคุณก่อน

- **1. การติดตั้ง Arduino IDE และ ESP32 Core:**
  - **ดาวน์โหลดและติดตั้งโปรแกรม Arduino IDE (เวอร์ชัน 2.x แนะนำ)**
  - **https://www.arduino.cc/en/software/**

เปิด Arduino IDE ไปที่ File > Preferences และเพิ่ม URL ต่อไปนี้ลงในช่อง "Additional boards manager URLs":

```
None
```

```
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json
```

ไปที่ Tools > Board > Boards Manager..., ค้นหาคำว่า `esp32` และทำการติดตั้ง

    - ตั้งค่าบอร์ดใน Arduino IDE โดยไปที่ Tools > Board และเลือก "ESP32C3 Dev Module"
- 2. การติดตั้งไดรเวอร์ USB:
    - บอร์ดพัฒนาส่วนใหญ่ต้องการไดรเวอร์เพื่อให้คอมพิวเตอร์มองเห็น โดยทั่วไปจะเป็นไดรเวอร์ CH340 หรือ CP210x หากคอมพิวเตอร์ของคุณมองไม่เห็นพอร์ตของบอร์ด ให้ลองค้นหาและติดตั้งไดรเวอร์เหล่านี้
- 3. การทดสอบเชื่อมต่อบอร์ด (Blink Test):
    - เสียบบอร์ด Senses-Hub เข้ากับคอมพิวเตอร์ผ่านสาย USB
    - ใน Arduino IDE ไปที่ Tools > Port และเลือกพอร์ตที่ปรากฏขึ้นมา
    - คัดลอกโค้ดด้านล่างนี้ไปวางในหน้าต่าง Arduino IDE:

```
C/C++
```

```
// กำหนดขาของ LED บนบอร์ด ESP32-C3
#define LED_BUILTIN 8
```

```
void setup() {
  // ตั้งค่าให้ Pin 8 เป็น Output
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // เปิด LED
  delay(1000);                       // หน่วงเวลา 1 วินาที
  digitalWrite(LED_BUILTIN, LOW);    // ปิด LED
  delay(1000);                       // หน่วงเวลา 1 วินาที
}
```

กดปุ่ม Upload (ลูกศรชี้ไปทางขวา) เพื่ออัปโหลดโค้ดลงบอร์ด

- **หากทุกอย่างถูกต้อง คุณจะเห็นไฟ LED บนบอร์ด ESP32-C3 กระพริบ ซึ่งหมายความว่าคุณพร้อมสำหรับบทต่อไปแล้ว!**

บทที่ 2: การเขียนโปรแกรมควบคุมบอร์ดและเซ็นเซอร์ (Programming the Board)

หลังจากที่เราเตรียมเครื่องมือและทดสอบการเชื่อมต่อบอร์ดสำเร็จแล้ว ในบทนี้เราจะมาลงลึกถึงการเขียนโปรแกรมเพื่อควบคุมส่วนต่างๆ ของบอร์ด Senses-Hub ตั้งแต่พื้นฐานการเขียนโค้ดสำหรับ ESP32, การดึงค่าจากเซ็นเซอร์ BME280 และ GY-521, ไปจนถึงการแสดงผลบนจอ OLED และปิดท้ายด้วยโปรเจกต์ย่อยเพื่อนำความรู้ทั้งหมดมารวมกัน

2.1 พื้นฐานการเขียนโค้ดสำหรับ ESP32

โค้ดใน Arduino IDE มีโครงสร้างหลักที่เข้าใจง่าย ประกอบด้วย 2 ฟังก์ชันสำคัญคือ `setup()` และ `loop()`

- **`void setup()`: ฟังก์ชันนี้จะทำงาน เพียงครั้งเดียว เมื่อบอร์ดเริ่มทำงาน (เปิดเครื่องหรือกดปุ่มรีเซ็ต) เรามักจะใช้ฟังก์ชันนี้ในการตั้งค่าเริ่มต้นต่างๆ เช่น กำหนดโหมดการทำงานของขา Pin, เริ่มการสื่อสารแบบ Serial, หรือเริ่มต้นการทำงานของเซ็นเซอร์**
- **`void loop()`: หลังจากฟังก์ชัน `setup()` ทำงานเสร็จ โปรแกรมจะเข้ามาทำงานในฟังก์ชันนี้วนซ้ำไปเรื่อยๆ ไม่สิ้นสุด ตราบใดที่บอร์ดยังมีไฟเลี้ยงอยู่ ที่นี่คือส่วนที่เราจะใส่โค้ดหลักของโปรแกรม เช่น การอ่านค่าจากเซ็นเซอร์, การแสดงผล, หรือการส่งข้อมูล**

การใช้งาน Serial Monitor สำหรับตรวจสอบการทำงาน (Debugging)

Serial Monitor คือเครื่องมือที่สำคัญที่สุดในการพัฒนโปรแกรม มันช่วยให้เราสามารถ "มองเห็น" ว่าเกิดอะไรขึ้นในบอร์ด โดยการพิมพ์ข้อความหรือค่าตัวแปรต่างๆ ออกมาทางพอร์ต USB

ตัวอย่างโค้ด:

C/C++

```
void setup() {
  // เริ่มการสื่อสารแบบ Serial ที่ความเร็ว 115200 bps
  Serial.begin(115200);
  Serial.println("Board setup complete. Starting loop...");
}

void loop() {
```

```
  // พิมพ์ข้อความออกมาทุกๆ 2 วินาที
  Serial.println("Hello from the loop function!");
  delay(2000); // หน่วงเวลา 2 วินาที
}
```

หลังจากอัปโหลดโค้ดนี้ ให้ไปที่ Tools > Serial Monitor เพื่อดูข้อความที่บอร์ดส่งออกมา

[ภาพหน้าจอ Serial Monitor แสดงข้อความ]

2.2 การอ่านค่าจากเซ็นเซอร์

ก่อนจะเริ่มอ่านค่าจากเซ็นเซอร์ เราต้องติดตั้ง "Library" ซึ่งเป็นชุดโค้ดสำเร็จรูปที่ช่วยให้เราสื่อสารกับเซ็นเซอร์ได้ง่ายขึ้น

การติดตั้ง Library:

1. **เปิด Arduino IDE ไปที่ Sketch > Include Library > Manage Libraries...**
2. **ค้นหาและติดตั้ง Library ต่อไปนี้:**
    - Adafruit BME280 Library
    - Adafruit Unified Sensor **(มักจะถูกติดตั้งมาพร้อมกับ BME280)**
    - Adafruit MPU6050
    - Adafruit GFX Library
    - Adafruit SSD1306

BME280: การอ่านค่าอุณหภูมิ, ความชื้น, และความกดอากาศ

เซ็นเซอร์ BME280 บนบอร์ดเชื่อมต่อผ่าน I2C ซึ่งเป็นรูปแบบการสื่อสารที่ใช้สายเพียง 2 เส้น (SDA และ SCL)

โค้ดตัวอย่าง:

C/C++

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
```

```
#include <Adafruit_BME280.h>

Adafruit_BME280 bme; // สร้าง object สำหรับ BME280

void setup() {
  Serial.begin(115200);
  Serial.println("BME280 Test");

  if (!bme.begin(0x76)) { // 0x76 คือ I2C address ของ BME280
    Serial.println("Could not find a valid BME280 sensor,
check wiring!");
    while (1);
  }
}

void loop() {
  Serial.print("Temperature = ");
  Serial.print(bme.readTemperature());
  Serial.println(" *C");

  Serial.print("Humidity = ");
  Serial.print(bme.readHumidity());
  Serial.println(" %");

  Serial.print("Pressure = ");
  Serial.print(bme.readPressure() / 100.0F);
  Serial.println(" hPa");
```

```
  Serial.println();
  delay(2000);
}
```

GY-521 (MPU-6050): การอ่านค่าการเคลื่อนไหว

เซ็นเซอร์นี้ก็เชื่อมต่อผ่าน I2C เช่นกัน เราสามารถใช้มันเพื่อตรวจจับการสั่น, การเอียง, และการเคลื่อนที่ได้

แนวคิดการประยุกต์ใช้: การตรวจจับการสั่นสะเทือนของเครื่องจักร (Vibration Monitoring)

เราสามารถนำค่าความเร่ง (Accelerometer) จากทั้ง 3 แกน (X, Y, Z) มาคำนวณหาขนาดของการสั่นสะเทือน หากค่านี้สูงเกินเกณฑ์ที่กำหนด อาจหมายถึงเครื่องจักรกำลังทำงานผิดปกติ

โค้ดตัวอย่าง:

C/C++

```
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>

Adafruit_MPU6050 mpu;

void setup(void) {
  Serial.begin(115200);
  Serial.println("MPU6050 Test");

  if (!mpu.begin()) {
```

```
    Serial.println("Failed to find MPU6050 chip");
    while (1);
  }
  Serial.println("MPU6050 Found!");
}

void loop() {
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);

  /* พิมพ์ค่าความเร่ง (Accelerometer) */
  Serial.print("Accel X: ");
Serial.print(a.acceleration.x);
  Serial.print(", Y: "); Serial.print(a.acceleration.y);
  Serial.print(", Z: "); Serial.println(a.acceleration.z);

  /* พิมพ์ค่าการหมุน (Gyroscope) */
  Serial.print("Gyro X: "); Serial.print(g.gyro.x);
  Serial.print(", Y: "); Serial.print(g.gyro.y);
  Serial.print(", Z: "); Serial.println(g.gyro.z);

  Serial.println("");
  delay(500);
}
```

2.3 การแสดงผลบนจอ OLED

จอ OLED ขนาดเล็กบนบอร์ดเป็นวิธีที่ยอดเยี่ยมในการแสดงข้อมูลโดยไม่ต้องต่อคอมพิวเตอร์

โค้ดตัวอย่าง:

C/C++

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128 // ความกว้างของจอ OLED
#define SCREEN_HEIGHT 64  // ความสูงของจอ OLED

// สร้าง object ของจอ โดยระบุขนาดและใช้ I2C
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT,
&Wire, -1);

void setup() {
  Serial.begin(115200);

  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // 0x3C
คือ I2C address ของจอ
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }
  display.display();
  delay(2000);

  // ล้างหน้าจอ
  display.clearDisplay();
```

```
  // ตั้งค่าข้อความ
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);

  // พิมพ์ข้อความ
  display.println("Hello, world!");
  display.display(); // แสดงผลที่ตั้งค่าไว้
}

void loop() {
  // ไม่ต้องทำอะไรใน loop สำหรับตัวอย่างนี้
}
```

2.4 โปรเจกต์ย่อย #1: สถานีตรวจวัดสภาพแวดล้อม

ถึงเวลานำความรู้ทั้งหมดมารวมกัน! โปรเจกต์นี้จะอ่านค่าจาก BME280 และ MPU-6050 แล้วนำมาแสดงผลบนจอ OLED

โค้ดฉบับสมบูรณ์:

```
None

#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_BME280.h>
#include <Adafruit_MPU6050.h>
```

```
// --- OLED Display Setup ---
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT,
&Wire, -1);

// --- Sensor Objects ---
Adafruit_BME280 bme;
Adafruit_MPU6050 mpu;

void setup() {
  Serial.begin(115200);

  // --- Initialize OLED ---
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }

  // --- Initialize BME280 ---
  if (!bme.begin(0x76)) {
    Serial.println("Could not find BME280 sensor");
    while (1);
  }

  // --- Initialize MPU6050 ---
  if (!mpu.begin()) {
```

```
    Serial.println("Failed to find MPU6050 chip");
    while (1);
  }

  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0,0);
  display.println("Sensors Initialized!");
  display.display();
  delay(2000);
}

void loop() {
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);

  // --- Clear display buffer ---
  display.clearDisplay();

  // --- Display BME280 Data ---
  display.setCursor(0, 0);
  display.print("Temp: ");
  display.print(bme.readTemperature());
  display.println(" C");
```

```
  display.print("Humi: ");
  display.print(bme.readHumidity());
  display.println(" %");

  // --- Display MPU6050 Data (Example: Z-axis
acceleration) ---
  display.setCursor(0, 30);
  display.print("Accel Z: ");
  display.println(a.acceleration.z);

  // --- Show buffer on display ---
  display.display();

  delay(500);
}
```

[ภาพถ่ายบอร์ด Senses-Hub แสดงค่าจากเซ็นเซอร์บนจอ OLED]

2.5 โปรเจกต์ย่อย #2: เครื่องตรวจจับการทำงานของเครื่องจักร

โปรเจกต์นี้จะเน้นการใช้ข้อมูลจาก GY-521 (MPU-6050) เพื่อจำแนกสถานะของเครื่องจักรอย่างง่ายๆ โดยดูจากขนาดของการสั่นสะเทือน

หลักการ:

1. อ่านค่าความเร่ง (ax, ay, az) จากเซ็นเซอร์
2. คำนวณขนาดของเวกเตอร์ความเร่งรวม (Magnitude) เพื่อดูการสั่นโดยไม่สนใจทิศทาง
3. เปรียบเทียบค่าที่ได้กับเกณฑ์ที่ตั้งไว้เพื่อตัดสินใจ

โค้ดตัวอย่าง (แสดงผลทาง Serial Monitor):

None

```
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>

Adafruit_MPU6050 mpu;

// --- Thresholds for machine state ---
const float VIBRATION_THRESHOLD = 2.5; // ค่าสำหรับแจ้งเตือนการสั่นที่รุนแรง
const float IDLE_THRESHOLD = 1.0;      // ค่าสำหรับสถานะปกติ/หยุดนิ่ง

void setup(void) {
  Serial.begin(115200);
  if (!mpu.begin()) {
    Serial.println("Failed to find MPU6050 chip");
    while (1);
  }
  Serial.println("Machine Vibration Monitor Initialized");
}

void loop() {
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);

  // คำนวณขนาดของการสั่น (Vector Magnitude)
```

```
  float total_accel = sqrt(pow(a.acceleration.x, 2) +
pow(a.acceleration.y, 2) + pow(a.acceleration.z, 2));

  Serial.print("Vibration Magnitude: ");
  Serial.print(total_accel);
  Serial.print(" | Status: ");

  // ตรวจสอบสถานะ
  if (total_accel > VIBRATION_THRESHOLD) {
    Serial.println("VIBRATION ALERT!");
  } else if (total_accel > IDLE_THRESHOLD) {
    Serial.println("RUNNING");
  } else {
    Serial.println("IDLE");
  }

  delay(200);
}
```

ในบทต่อไป เราจะนำข้อมูลที่ได้จากโปรเจกต์เหล่านี้ส่งเข้าไปยัง n8n เพื่อเริ่มต้นการสร้างระบบอัตโนมัติกันครับ

บทที่ 3: รู้จักกับ n8n แพลตฟอร์มเชื่อมทุกอย่างเข้าด้วยกัน (Introduction to n8n)

ในบทที่แล้ว เราได้เรียนรู้วิธีการดึงข้อมูลจากโลกจริงผ่านเซ็นเซอร์บนบอร์ด Senses-Hub ได้สำเร็จ ตอนนี้ถึงเวลาที่จะนำข้อมูลเหล่านั้นมาทำให้เกิดประโยชน์แล้ว และเครื่องมือที่เราจะใช้ก็คือ n8n (อ่านว่า "เอ็น-เอท-เอ็น") ซึ่งเป็นแพลตฟอร์ม Workflow Automation ที่ทรงพลังและยืดหยุ่นอย่างยิ่ง

ในบทนี้ เราจะมาทำความรู้จักว่า n8n คืออะไร, ทำไมมันถึงเป็นเครื่องมือที่ยอดเยี่ยมสำหรับโปรเจกต์ของเรา, วิธีการติดตั้งแบบ Local บนคอมพิวเตอร์ของคุณ, และทดลองสร้าง Workflow แรกเพื่อทำความเข้าใจหลักการทำงานเบื้องต้น

3.1 n8n คืออะไร?

ลองจินตนาการว่าคุณมีตัวต่อ LEGO จำนวนมาก แต่ละชิ้นมีความสามารถแตกต่างกันไป เช่น ชิ้นหนึ่งคือ "รับอีเมล", อีกชิ้นคือ "บันทึกข้อมูลลง Google Sheets", และอีกชิ้นคือ "ส่งข้อความแจ้งเตือนผ่าน LINE"

n8n ก็เปรียบเสมือนแผ่น LEGO ขนาดใหญ่ (Workflow) ที่ให้คุณนำตัวต่อ (เรียกว่า Node) เหล่านี้มาประกอบกันเป็นเรื่องราวหรือกระบวนการทำงานอัตโนมัติตามที่คุณต้องการ โดยไม่ต้องเขียนโค้ดที่ซับซ้อน

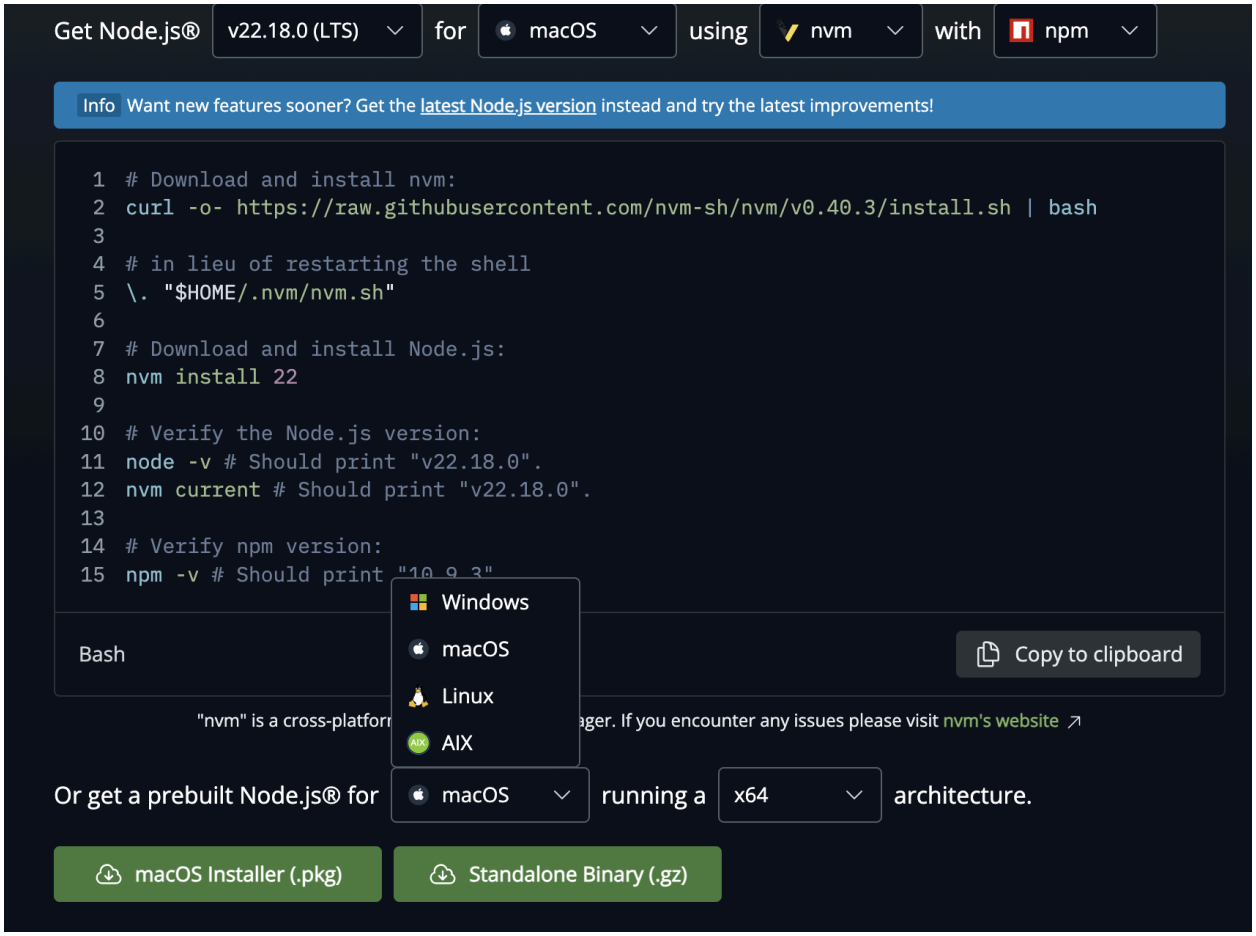
แนวคิดสำคัญคือ Low-code/No-code ซึ่งหมายความว่าเราสามารถสร้างระบบที่ซับซ้อนได้โดยการลากและวาง (Drag-and-Drop) Node ต่างๆ แล้วตั้งค่าการทำงานเพียงเล็กน้อยเท่านั้น ทำให้การเชื่อมต่อระหว่างแอปพลิเคชัน, API, และอุปกรณ์ IoT กลายเป็นเรื่องง่าย

3.2 การติดตั้งและเริ่มต้นใช้งาน n8n

สำหรับโปรเจกต์นี้ เราจะเน้นการติดตั้ง n8n บนคอมพิวเตอร์ของเราเอง (Local) เพื่อให้สามารถทำงานได้โดยไม่ต้องเชื่อมต่ออินเทอร์เน็ตตลอดเวลาและมีความเป็นส่วนตัวสูง เราจะใช้วิธีการติดตั้งผ่าน npm (Node Package Manager) ซึ่งเป็นวิธีมาตรฐานและตรงไปตรงมา

ขั้นตอนการติดตั้งแบบ Local บนคอมพิวเตอร์:

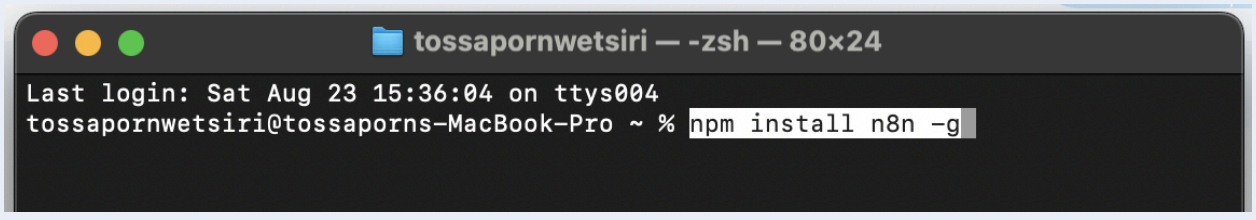
1. **ติดตั้ง Node.js: n8n ทำงานบน Node.js ดังนั้นคุณต้องติดตั้งก่อน**
    - **ดาวน์โหลดและติดตั้ง Node.js (แนะนำเวอร์ชัน LTS) จาก https://nodejs.org/en/download**
    - **npm จะถูกติดตั้งมาพร้อมกับ Node.js โดยอัตโนมัติ**



2. **เปิดโปรแกรม Terminal (หรือ Command Prompt/PowerShell บน Windows):**
3. **รันคำสั่งเพื่อติดตั้ง n8n: คัดลอกและวางคำสั่งด้านล่างนี้ลงใน Terminal แล้วกด Enter**
   การติดตั้งนี้อาจใช้เวลาสักครู่

None

```
npm install n8n -g
```

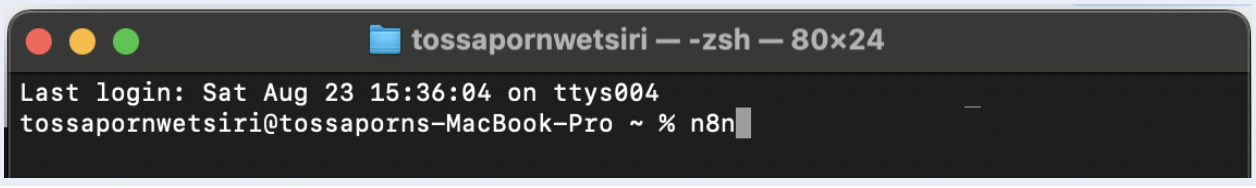


`npm install n8n`: เป็นคำสั่งให้ติดตั้งแพ็กเกจ n8n

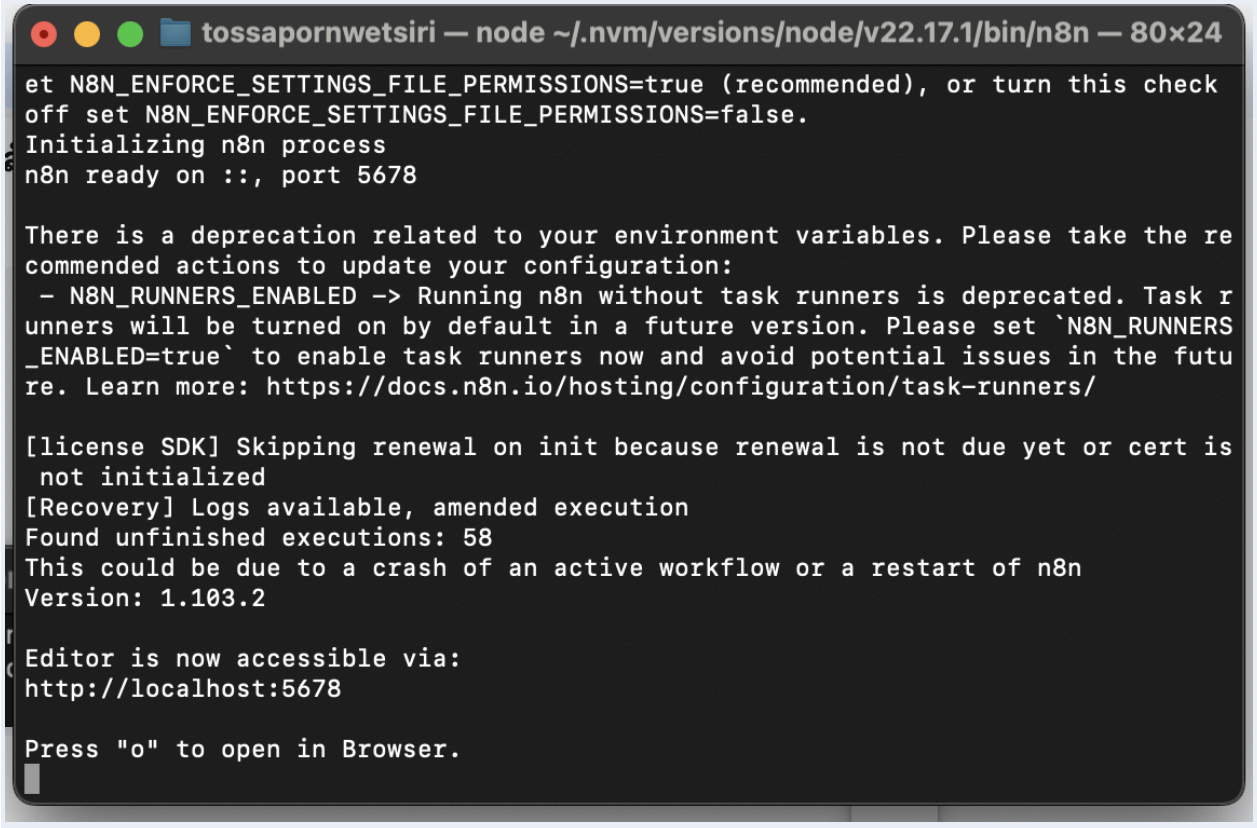
- **`-g`: หมายถึงการติดตั้งแบบ Global ทำให้คุณสามารถเรียกใช้คำสั่ง `n8n` จากที่ไหนก็ได้ในเครื่องของคุณ**

4. **เริ่มต้น n8n: หลังจากติดตั้งเสร็จ พิมพ์คำสั่งต่อไปนี้ใน Terminal เพื่อเริ่มการทำงานของ n8n:**

```
None
```



รอสักครู่: เมื่อ n8n เริ่มทำงานเรียบร้อยแล้ว คุณจะเห็นข้อความว่า `Editor is now available on http://localhost:5678`



5. **เปิดเว็บเบราว์เซอร์: เข้าไปที่ `http://localhost:5678` คุณจะพบกับหน้าจอของ n8n พร้อมใช้งาน!**

[ภาพหน้าจอ n8n UI ครั้งแรก]

3.3 ส่วนประกอบหลักของ n8n ที่ต้องรู้

เมื่อเข้ามาใน n8n คุณจะพบกับหน้าจอหลักที่มีส่วนประกอบสำคัญดังนี้:

- **Workflow: คือพื้นที่ทำงานทั้งหมด เปรียบเสมือนผืนผ้าใบที่คุณจะใช้วาดกระบวนการทำงานอัตโนมัติ**
- **Node: คือกล่องการทำงานแต่ละอย่าง แบ่งเป็น 2 ประเภทหลักๆ**
  - **Trigger Node: Node เริ่มต้นของ Workflow เป็นตัวกำหนดว่าจะให้ Workflow นี้เริ่มทำงาน "เมื่อไหร่" หรือ "เพราะอะไร" (เช่น เมื่อมีคนส่งข้อมูลเข้ามา, เมื่อถึงเวลาที่กำหนด)**
  - **Regular/Action Node: Node ที่ทำงานต่างๆ หลังจาก Trigger เริ่มทำงาน (เช่น บันทึกข้อมูล, ส่งอีเมล, เรียกใช้ AI)**
- **Connection: คือเส้นที่เชื่อมระหว่าง Node เพื่อบอกทิศทางการไหลของข้อมูลจาก Node หนึ่งไปยังอีก Node หนึ่ง**
- **Credential: คือที่สำหรับเก็บข้อมูลสำคัญและเป็นความลับ เช่น API Keys,**

Username/Password เพื่อให้ n8n สามารถเชื่อมต่อกับบริการอื่นๆ ได้อย่างปลอดภัย

[ภาพหน้าจอ n8n UI พร้อมคำอธิบายส่วนประกอบต่างๆ]

3.4 สร้าง Workflow แรก: "Hello, n8n!"

เพื่อทำความเข้าใจการไหลของข้อมูล เราจะมาสร้าง Workflow ที่ง่ายที่สุดกัน โดยจะให้มันเริ่มทำงานเมื่อเรากดปุ่ม และให้มันสร้างข้อความ "Hello, n8n!" ขึ้นมา

ขั้นตอน:

1. สร้าง Workflow ใหม่: ที่หน้าจอหลักของ n8n กดปุ่ม `Add workflow`
2. เริ่มต้นด้วย Manual Trigger: โดยปกติ n8n จะมี Node `Start` มาให้ ซึ่งเป็น Manual Trigger อยู่แล้ว Node นี้หมายความว่า Workflow จะเริ่มทำงานก็ต่อเมื่อเรากดปุ่ม "Execute Workflow"
3. เพิ่ม Node ใหม่: กดที่เครื่องหมาย `+` ที่ต่อท้ายจาก Node `Start`
4. ค้นหา Set Node: ในช่องค้นหา พิมพ์คำว่า `Set` แล้วเลือก Node ที่ชื่อว่า `Set`
    - Set Node เป็น Node ที่มีประโยชน์มาก ใช้สำหรับสร้าง, แก้ไข, หรือจัดรูปแบบข้อมูล
5. ตั้งค่า Set Node:
    - คลิกที่ Node `Set` ที่เพิ่งเพิ่มเข้ามา
    - ในช่อง Name ให้พิมพ์ `message`
    - ในช่อง Value ให้พิมพ์ `Hello, n8n!`
    - ตรวจสอบว่าตัวเลือก Keep Only Set ถูก ปิด อยู่ (เพื่อให้เราเห็นข้อมูลจาก Node ก่อนหน้าด้วย)
6. ทดสอบการทำงาน:
    - กดปุ่ม Execute Workflow ที่มุมขวาล่าง
    - รอสักครู่ Node ทั้งสองจะทำงานเสร็จและมีกรอบสีเขียว
    - คลิกที่ Set Node คุณจะเห็นผลลัพธ์ในฝั่ง Output ว่ามีข้อมูล `message` ที่มีค่าเป็น `Hello, n8n!` เพิ่มเข้ามาเรียบร้อยแล้ว

[ภาพ GIF แสดงขั้นตอนการสร้าง Workflow "Hello, n8n!"]

ยอดเยี่ยม! ตอนนี้คุณได้เข้าใจหลักการทำงานพื้นฐานของ n8n แล้ว ในบทต่อไป เราจะทำการเชื่อมต่อบอร์ด Senses-Hub ของเราเพื่อส่งข้อมูลเซ็นเซอร์เข้ามาใน n8n กันครับ

บทที่ 4: การเชื่อมต่อบอร์ด IoT กับ n8n

(Connecting IoT to n8n)

ในบทนี้ เราจะสร้างสะพานเชื่อมระหว่างบอร์ด Senses-Hub กับ n8n โดยใช้วิธีที่ง่ายและเหมาะสำหรับงาน IoT โดยเฉพาะ นั่นคือการใช้แพลตฟอร์ม Sensesiot.net

Sensesiot.net คืออะไร?

Sensesiot คือแพลตฟอร์ม IoT ที่ทำหน้าที่เป็นตัวกลางอัจฉริยะ ช่วยให้เราส่งข้อมูลจากอุปกรณ์ (เช่น บอร์ด Senses-Hub) ขึ้นไปบนคลาวด์ได้อย่างง่ายดาย และที่สำคัญคือ Sensesiot สามารถทำงานร่วมกับ n8n ได้อย่างราบรื่น ทำให้เราไม่ต้องจัดการกับการเชื่อมต่อที่ซับซ้อนอย่าง Webhook หรือ MQTT Broker ด้วยตัวเอง

สถาปัตยกรรมใหม่ของเรา:

Hardware (Senses-Hub) ⟶ Sensesiot.net (ตัวกลาง) ⟶ n8n (Automation)

4.1 การเตรียมความพร้อมฝั่ง Arduino

การติดตั้ง Library Sensesiot

ก่อนที่เราจะเขียนโค้ดได้ เราต้องติดตั้ง Library เพื่อให้บอร์ดของเราสามารถสื่อสารกับ Sensesiot ได้

ขั้นตอนการติดตั้ง:

1. **เปิดโปรแกรม Arduino IDE**
2. **ไปที่เมนู Sketch > Include Library > Manage Libraries... (หรือกด `Ctrl+Shift+I`)**
3. **ในช่องค้นหา พิมพ์คำว่า `sensesiot`**
4. **คุณจะพบ Library ชื่อ Sensesiot by Natthawat Raocharoensinp...**
5. **คลิกปุ่ม Install เพื่อทำการติดตั้ง**

[ภาพหน้าจอ Library Manager แสดงการติดตั้ง Sensesiot]

การเตรียม Device Key จาก Sensesiot.net

1. **ไปที่เว็บไซต์ sensesiot.net และทำการสมัครสมาชิก/เข้าสู่ระบบ**
2. **สร้าง Device ใหม่ และตั้งค่า Slot ของข้อมูลที่คุณต้องการส่ง (เช่น Slot 1: Temperature, Slot 2: Humidity, Slot 3: Vibration)**
3. **หลังจากสร้าง Device แล้ว ให้คัดลอก Device Key มาเก็บไว้ เพราะเราจะต้องใช้ในโค้ด Arduino**

4.2 การส่งข้อมูลจาก ESP32 สู่ Sensesiot.net

ตอนนี้เราจะเขียนโค้ดเพื่ออ่านค่าจากเซ็นเซอร์ BME280 และ GY-521 (เฉพาะการสั่น) แล้วส่งไปยัง

Sensesiot ผ่าน Library ที่เราเพิ่งติดตั้งไป

สำคัญ: แก้ไข `wifissid`, `wifipw`, และ `key` ให้เป็นของคุณ

None

```
/*
  Example - Publish BME280 & MPU6050 Data to Sensesiot Platform
  Adapted for Senses-Hub SH-BC3
*/
#include <Sensesiot.h>
#include <Adafruit_BME280.h>
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>

// --- Sensesiot & WiFi Credentials ---
const char key[] = "YOUR_SENSESIOT_DEVICE_KEY"; // <<<< ใส่ Device Key ที่คัดลอกมา
const char wifissid[] = "YOUR_WIFI_SSID";         // <<<< ใส่ชื่อ Wi-Fi ของคุณ
const char wifipw[] = "YOUR_WIFI_PASSWORD";     // <<<< ใส่รหัสผ่าน Wi-Fi ของคุณ

SensesiotClient sensesProtocol(key);

// --- Sensor Objects ---
Adafruit_BME280 bme;
Adafruit_MPU6050 mpu;
```

```
void setup()
{
  Serial.begin(115200);

  // --- Initialize Sensors ---
  if (!bme.begin(0x76)) {
    Serial.println("Could not find BME280 sensor");
    while (1);
  }
  if (!mpu.begin()) {
    Serial.println("Failed to find MPU6050 chip");
    while (1);
  }
  Serial.println("Sensors Initialized.");

  // --- Connect to WiFi & Sensesiot ---
  sensesProtocol.begin(wifissid, wifipw);
  sensesProtocol.waitUntilReady();
  Serial.println(F("Connected to Sensesiot Platform."));
}

void loop()
{
  // --- Check Connection ---
  if (!sensesProtocol.ready())
```

```
  {
    Serial.println(F("Disconnected. Reconnecting..."));
    sensesProtocol.begin(wifissid, wifipw);
    sensesProtocol.waitUntilReady();
    Serial.println(F("Reconnected."));
  }
  sensesProtocol.loop(); // Must be called in loop to
maintain connection


  // --- Read Sensor Data ---
  float temperature = bme.readTemperature();
  float humidity = bme.readHumidity();


  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);
  // คำนวณขนาดการสั่นสะเทือน
  float vibration = sqrt(pow(a.acceleration.x, 2) +
pow(a.acceleration.y, 2) + pow(a.acceleration.z, 2));


  // --- Send Data to Sensesiot Slots ---
  sensesProtocol.setData(1, temperature); // ส่งค่าอุณหภูมิไปที่
Slot 1
  sensesProtocol.setData(2, humidity);    // ส่งค่าความชื้นไปที่
Slot 2
  sensesProtocol.setData(3, vibration);   // ส่งค่าการสั่นไปที่
Slot 3


  Serial.print("Sending Data -> Temp: ");
```

```
  Serial.print(temperature);
  Serial.print(" C, Humi: ");
  Serial.print(humidity);
  Serial.print(" %, Vib: ");
  Serial.println(vibration);

  delay(10000); // ส่งข้อมูลทุกๆ 10 วินาที
}
```

ทดสอบการทำงาน:

อัปโหลดโค้ดนี้ลงบอร์ด Senses-Hub จากนั้นไปที่หน้า Dashboard ของ Device ใน Sensesiot.net คุณจะเห็นข้อมูลจากเซ็นเซอร์ถูกส่งเข้ามาแบบ Real-time

4.3 การรับข้อมูลจาก Sensesiot ใน n8n

เมื่อข้อมูลของเราอยู่บนแพลตฟอร์ม Sensesiot แล้ว การนำข้อมูลเข้ามาใน n8n จะกลายเป็นเรื่องง่ายมาก

ขั้นตอนใน n8n:

1. **เพิ่ม Sensesiot Trigger Node: สร้าง Workflow ใหม่ และเพิ่ม Node โดยค้นหาคำว่า `Sensesiot` คุณจะพบ `Sensesiot Trigger`**
2. **ตั้งค่า Credential:**
    - **ในช่อง Sensesiot API Credential, เลือก `Create New`**
    - **คุณจะต้องใช้ API Key จาก Sensesiot.net (สามารถหาได้จากหน้า Profile ของคุณ)**
    - **นำ API Key มาใส่ แล้วกด Save**
3. **เลือก Device และ Slot:**
    - **Device: เลือก Device ที่คุณสร้างไว้จาก Dropdown list**
    - **Slot: เลือก Slot ของข้อมูลที่คุณต้องการให้เป็นตัวกระตุ้น Workflow นี้ (เช่น เลือก Slot 1 เพื่อให้ Workflow ทำงานทุกครั้งที่มีค่าอุณหภูมิใหม่เข้ามา)**
4. **เริ่มรอรับข้อมูล: กด Listen for Test Event**
5. **เมื่อบอร์ดของคุณส่งข้อมูลใหม่เข้ามาที่ Sensesiot, Node ใน n8n ก็จะได้รับข้อมูลนั้นทันที!**

[ภาพหน้าจอ Workflow แสดงการตั้งค่า Sensesiot Trigger Node]

ยอดเยี่ยม! ตอนนี้เราได้สร้างการเชื่อมต่อที่สมบูรณ์และมีประสิทธิภาพจากฮาร์ดแวร์ไปยัง n8n โดยมี Sensesiot เป็นตัวกลางที่เชื่อถือได้ ในบทต่อไป เราจะนำข้อมูลที่ได้นี้ไปจัดเก็บลงในฐานข้อมูล PostgreSQL กันครับ

บทที่ 5: การจัดเก็บและจัดการข้อมูลด้วย PostgreSQL (Data Persistence with

PostgreSQL)

ในบทที่แล้ว เราประสบความสำเร็จในการนำข้อมูลจากบอร์ด Senses-Hub เข้ามาใน n8n ได้แล้ว แต่ข้อมูลที่เข้ามาใน Workflow จะหายไปเมื่อ Workflow ทำงานเสร็จสิ้น หากเราต้องการนำข้อมูลไปวิเคราะห์ย้อนหลัง, สร้างกราฟ, หรือสอน AI เราจำเป็นต้องมีที่สำหรับ "จัดเก็บ" ข้อมูลเหล่านั้นอย่างถาวร

ในบทนี้ เราจะใช้ PostgreSQL ซึ่งเป็นระบบฐานข้อมูลแบบ Open-source ที่ทรงพลังและได้รับความนิยมสูง มาทำหน้าที่เป็นคลังข้อมูล (Data Warehouse) ให้กับโปรเจกต์ของเรา

5.1 ทำไมต้องมีฐานข้อมูล?

การมีฐานข้อมูลให้ประโยชน์หลายอย่าง:

- **การจัดเก็บข้อมูลถาวร (Persistence): ข้อมูลเซ็นเซอร์จะถูกเก็บไว้อย่างปลอดภัยและไม่สูญหาย**
- **การวิเคราะห์ข้อมูลย้อนหลัง: เราสามารถดึงข้อมูลในอดีตมาดูแนวโน้มได้ เช่น อุณหภูมิเฉลี่ยในสัปดาห์ที่ผ่านมา**
- **พื้นฐานสำหรับ AI: เป็นแหล่งข้อมูลสำคัญให้ AI Agent ของเราได้เรียนรู้และนำไปใช้ในการตัดสินใจ**
- **ความเป็นระเบียบ: ข้อมูลจะถูกจัดเก็บในรูปแบบตาราง (Table) ที่มีโครงสร้างชัดเจน ทำให้ง่ายต่อการจัดการและค้นหา**

5.2 การติดตั้ง PostgreSQL เบื้องต้น

เช่นเดียวกับ n8n วิธีที่ง่ายและสะดวกที่สุดในการติดตั้ง PostgreSQL บนเครื่องคอมพิวเตอร์ของเราคือการใช้ Docker

ขั้นตอนการติดตั้ง:

1. **ตรวจสอบว่า Docker Desktop กำลังทำงานอยู่**
2. **เปิดโปรแกรม Terminal (หรือ Command Prompt/PowerShell บน Windows)**
3. **รันคำสั่งเพื่อสร้างและเปิดใช้งาน PostgreSQL Container: คัดลอกและวางคำสั่งด้านล่างนี้ลงใน Terminal แล้วกด Enter**

None

4.

```
docker run --name my-postgres -e
POSTGRES_PASSWORD=mysecretpassword -p 5432:5432 -d postgres
```

5.

    - `--name my-postgres`: ตั้งชื่อ Container ของเราว่า `my-postgres`
    - `-e POSTGRES_PASSWORD=mysecretpassword`: ตั้งรหัสผ่านของฐานข้อมูลเป็น `mysecretpassword` (สำคัญ: ในการใช้งานจริงควรเปลี่ยนเป็นรหัสที่ซับซ้อนกว่านี้)
    - `-p 5432:5432`: เชื่อมพอร์ต 5432 ของคอมพิวเตอร์เราเข้ากับพอร์ตมาตรฐานของ PostgreSQL
    - `-d postgres`: บอกให้ Docker ใช้ Image ของ PostgreSQL และทำงานอยู่เบื้องหลัง (detached mode)
6. ตรวจสอบสถานะ: พิมพ์คำสั่ง `docker ps` คุณควรจะเห็น Container ชื่อ `my-postgres` กำลังทำงานอยู่

เพียงเท่านี้คุณก็มีเซิร์ฟเวอร์ฐานข้อมูล PostgreSQL พร้อมใช้งานบนเครื่องของคุณแล้ว!

5.3 การเชื่อมต่อ n8n กับ PostgreSQL

ตอนนี้เราจะมาตั้งค่าให้ n8n รู้จักกับฐานข้อมูลที่เราเพิ่งสร้างขึ้น

ขั้นตอนใน n8n:

1. ไปที่เมนูด้านซ้าย เลือก Credentials
2. คลิก Add Credential
3. ในช่องค้นหา พิมพ์ `Postgres` แล้วเลือก Postgres Credential
4. กรอกข้อมูลการเชื่อมต่อ:
    - Host: `localhost` (เพราะฐานข้อมูลทำงานอยู่บนเครื่องเดียวกับ n8n)
    - Database: `postgres` (เป็นชื่อฐานข้อมูลเริ่มต้น)
    - User: `postgres` (เป็นชื่อผู้ใช้เริ่มต้น)
    - Password: `mysecretpassword` (รหัสผ่านที่เราตั้งไว้ในคำสั่ง Docker)
    - Port: `5432`
5. ตั้งชื่อ Credential (เช่น `My Local Postgres`) แล้วกด Save

[ภาพหน้าจอการตั้งค่า Postgres Credential ใน n8n]

5.4 Workflow: บันทึกข้อมูลเซ็นเซอร์ลงฐานข้อมูล

เราจะสร้าง Workflow ที่สมบูรณ์ซึ่งรับข้อมูลจาก Sensesiot และบันทึกลงใน PostgreSQL โดยอัตโนมัติ

ขั้นตอนการสร้าง Workflow:

1. เริ่มต้นด้วย Sensesiot Trigger: สร้าง Workflow ใหม่และเพิ่ม Node `Sensesiot Trigger` ตั้งค่าให้รับข้อมูลจาก Device และ Slot ที่คุณต้องการ (เหมือนในบทที่ 4)
2. เพิ่ม Postgres Node: กด `+` ต่อจาก Sensesiot Trigger แล้วค้นหาและเพิ่ม Node

Postgres

3. **ตั้งค่า Postgres Node:**
    - **Credential: เลือก Credential ที่เราเพิ่งสร้าง (My Local Postgres)**
    - **Operation: เลือก Execute Query**
    - **Query: พิมพ์คำสั่ง SQL ต่อไปนี้ลงไป**

None

    -

```
-- สร้างตารางก่อน หากยังไม่มี
CREATE TABLE IF NOT EXISTS sensor_data (
    id SERIAL PRIMARY KEY,
    timestamp TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    temperature REAL,
    humidity REAL,
    vibration REAL
);

-- แทรกข้อมูลใหม่เข้าไป
INSERT INTO sensor_data (temperature, humidity, vibration)
VALUES (
    {{ $json.body.slot1 }},
    {{ $json.body.slot2 }},
    {{ $json.body.slot3 }}
);
```

    -

คำอธิบายคำสั่ง SQL:

- **CREATE TABLE IF NOT EXISTS ...: เป็นคำสั่งที่ปลอดภัย ใช้สร้างตารางชื่อ sensor_data หากตารางนี้ยังไม่มีอยู่ โดยมีคอลัมน์สำหรับเก็บ ID, เวลา, อุณหภูมิ, ความชื้น, และการสั่น**

- `INSERT INTO sensor_data ...`: เป็นคำสั่งสำหรับเพิ่มข้อมูลแถวใหม่เข้าไปในตาราง
- `{{ $json.body.slot1 }}`: นี่คือ Expression ของ n8n มันคือวิธีที่เราจะดึงข้อมูลที่ได้รับจาก Node ก่อนหน้า (Sensesiot Trigger) มาใช้งาน ในที่นี้คือการดึงค่าจาก `slot1`, `slot2`, และ `slot3` มาใส่ในคอลัมน์ที่ถูกต้อง

4. ทดสอบการทำงาน:
    - กด Listen for Test Event ที่ `Sensesiot Trigger`
    - เมื่อมีข้อมูลเข้ามา ให้กด Execute Node ที่ `Postgres Node`
    - หากไม่มีข้อผิดพลาด (Node เป็นสีเขียว) หมายความว่าข้อมูลถูกบันทึกลงฐานข้อมูลเรียบร้อยแล้ว!

[ภาพหน้าจอ Workflow ที่เชื่อมต่อ Sensesiot Trigger กับ Postgres Node]

การตรวจสอบข้อมูลในฐานข้อมูล (ทางเลือก):

คุณสามารถใช้โปรแกรมจัดการฐานข้อมูล เช่น DBeaver หรือ pgAdmin เพื่อเชื่อมต่อกับ PostgreSQL ที่รันบน Docker และเปิดดูข้อมูลในตาราง sensor_data เพื่อให้แน่ใจว่าข้อมูลถูกบันทึกอย่างถูกต้อง

บทนี้ถือเป็นก้าวสำคัญ เราได้สร้างระบบที่สามารถรวบรวมและจัดเก็บข้อมูล IoT ได้อย่างสมบูรณ์แล้ว ในบทสุดท้าย เราจะนำข้อมูลที่สะสมไว้ในฐานข้อมูลนี้มาสร้างเป็น AI Agent ผู้ช่วยอัจฉริยะกันครับ

บทที่ 6: สร้างผู้ช่วยอัจฉริยะ (AI Agent) ด้วย n8n

เราเดินทางมาถึงบทสุดท้าย ซึ่งเป็นส่วนที่น่าตื่นเต้นที่สุด เราได้สร้างระบบที่สามารถรวบรวมข้อมูลจากเซ็นเซอร์ (บทที่ 2), ส่งข้อมูลผ่านแพลตฟอร์ม IoT (บทที่ 4), และจัดเก็บข้อมูลลงฐานข้อมูลอย่างเป็นระบบ (บทที่ 5) ได้สำเร็จแล้ว ตอนนี้ถึงเวลาที่จะนำข้อมูลเหล่านั้นมา "ปลุกเสก" ให้มีชีวิตชีวาและมีความคิดเป็นของตัวเองด้วย Large Language Models (LLMs)

ในบทนี้ เราจะเรียนรู้วิธีการเชื่อมต่อ n8n กับ LLM ทั้งแบบ Cloud-based และแบบ Local ที่ทำงานบนเครื่องคอมพิวเตอร์ของเราเอง เพื่อสร้าง AI Agent ที่สามารถวิเคราะห์ข้อมูลและตอบคำถาม

ได้อย่างชาญฉลาด

6.1 หลักการทำงานของ AI Agent และ LLMs

- **Large Language Models (LLMs): คือแบบจำลองปัญญาประดิษฐ์ที่ถูกฝึกฝนด้วยข้อมูลข้อความมหาศาล ทำให้มันมีความสามารถในการเข้าใจ, สรุป, แปล, และสร้างข้อความใหม่ๆ ได้เหมือนมนุษย์ ตัวอย่างที่รู้จักกันดีคือ GPT-4 (ของ OpenAI) หรือ Llama (ของ Meta)**
- **AI Agent: ในบริบทของเรา คือ Workflow ใน n8n ที่เราออกแบบให้ทำงานร่วมกับ LLM โดยมีขั้นตอนคือ:**
    1. **รวบรวมข้อมูล (Gather Data): ดึงข้อมูลที่เกี่ยวข้องจากฐานข้อมูล PostgreSQL ของเรา**
    2. **สร้างคำสั่ง (Construct Prompt): นำข้อมูลที่ได้มาสร้างเป็น "คำสั่ง" หรือ "คำถาม" ที่ชัดเจน เพื่อบอกให้ LLM รู้ว่าเราต้องการให้มันทำอะไร**
    3. **เรียกใช้ LLM (Query LLM): ส่งคำสั่งนั้นไปให้ LLM ประมวลผล**
    4. **ดำเนินการต่อ (Take Action): นำคำตอบที่ได้จาก LLM ไปใช้งานต่อ เช่น ส่งเป็นรายงาน, แสดงผล, หรือใช้ในการตัดสินใจ**

6.2 การเชื่อมต่อ Large Language Models (LLM)

เรามีทางเลือกหลักๆ สองทางในการนำพลังของ LLM มาใช้ใน n8n

6.2.1 การใช้ Cloud-based LLM (เช่น OpenAI, Google Gemini)

เป็นวิธีที่ง่ายในการเริ่มต้น แต่มีค่าใช้จ่ายตามการใช้งานและต้องเชื่อมต่ออินเทอร์เน็ต

- **ขั้นตอน:**
    1. **สมัครใช้บริการและขอ API Key จากผู้ให้บริการ (เช่น platform.openai.com)**
    2. **ใน n8n, ไปที่ Credentials และสร้าง Credential สำหรับบริการนั้นๆ (เช่น `OpenAI API`)**
    3. **ใช้ Node สำเร็จรูป เช่น `OpenAI Chat Model` ใน Workflow ของคุณ**

6.2.2 การใช้ Local LLM (ทำงานบนเครื่องของคุณ)

วิธีนี้ให้ความเป็นส่วนตัวสูง, ไม่มีค่าใช้จ่ายในการเรียกใช้งาน, และทำงานได้แม้ออฟไลน์ เหมาะกับโปรแกรมของเราอย่างยิ่ง เราจะเน้นที่วิธีนี้เป็นหลัก โดยใช้เครื่องมือยอดนิยม 2 ตัวคือ Ollama และ LM Studio

ขั้นตอนการเตรียม Local LLM:

1. **ติดตั้ง Ollama หรือ LM Studio:**
    - **Ollama: (แนะนำสำหรับผู้เริ่มต้น) ไปที่ ollama.com ดาวน์โหลดและติดตั้งโปรแกรม**

    - LM Studio: ไปที่ lmstudio.ai ดาวน์โหลดและติดตั้งโปรแกรม
2. ดาวน์โหลดโมเดล:
    - Ollama: เปิด Terminal แล้วรันคำสั่ง `ollama run llama3` (หรือโมเดลอื่นที่คุณต้องการ เช่น `gemma`)
    - LM Studio: เปิดโปรแกรม, ไปที่หน้าค้นหา (แว่นขยาย), ค้นหาโมเดลที่ต้องการ (เช่น Llama-3), แล้วกดดาวน์โหลด
3. เปิดใช้งาน API Server:
    - Ollama: เมื่อคุณรันคำสั่งในข้อ 2, Ollama จะเปิด API Server ที่ `http://localhost:11434` ให้โดยอัตโนมัติ
    - LM Studio: ไปที่หน้า Local Server (รูป `<->`), เลือกโมเดลที่ดาวน์โหลดมา, แล้วกด Start Server โปรแกรมจะสร้าง API endpoint ที่ `http://localhost:1234/v1/` ซึ่งเข้ากันได้กับ API ของ OpenAI

การเชื่อมต่อ n8n กับ Local LLM:

1. เพิ่ม Ollama Node: ใน n8n, ค้นหาและเพิ่ม Node `Ollama`
2. ตั้งค่า Credential:
    - เลือก `Create New`
    - Base URL: ใส่ `http://localhost:11434` (สำหรับ Ollama) หรือ `http://localhost:1234/v1` (สำหรับ LM Studio)
    - กด Save
3. ตั้งค่า Node:
    - Model: ใส่ชื่อโมเดลที่คุณรันอยู่ (เช่น `llama3`)
    - Prompt: ใส่คำสั่งที่คุณต้องการให้ AI ทำ

6.3 ตัวอย่าง Workflow สำหรับ AI Agent

AI Agent #1: นักวิเคราะห์ข้อมูลสภาพอากาศและการสั่นสะเทือน

Workflow นี้จะทำงานทุกๆ ชั่วโมง เพื่อสรุปข้อมูลจากฐานข้อมูลแล้วส่งเป็นรายงาน

โครงสร้าง Workflow: `Schedule Trigger` -> `Postgres (Read)` -> `Ollama` -> `(Optional) Email/LINE`

1. Schedule Trigger: ตั้งค่าให้ทำงานทุก 1 ชั่วโมง (`Interval`)
2. Postgres Node:
    - Operation: `Execute Query`
    - Query:

None

```
SELECT * FROM sensor_data
WHERE timestamp >= NOW() - INTERVAL '1 hour'
ORDER BY timestamp DESC;
```

(ดึงข้อมูลทั้งหมดในชั่วโมงล่าสุด)

3. **Ollama Node:**
   - **Prompt:**

None

```
คุณคือผู้ช่วยวิเคราะห์ข้อมูลจากเซ็นเซอร์ IoT นี่คือข้อมูลดิบในรูปแบบ JSON จากชั่วโมงที่ผ่านมา:

{{ JSON.stringify($input.all()) }}

โปรดสรุปข้อมูลทั้งหมดนี้เป็นภาษาไทยที่เข้าใจง่าย โดยมีหัวข้อดังนี้:
1. สรุปอุณหภูมิ: ค่าสูงสุด, ต่ำสุด, และแนวโน้ม
2. สรุปความชื้น: ค่าสูงสุด, ต่ำสุด, และแนวโน้ม
3. สรุปการสั่นสะเทือน: มีการสั่นที่ผิดปกติหรือไม่? ถ้ามี ให้ระบุช่วงเวลา
4. คำแนะนำ: จากข้อมูลทั้งหมด มีอะไรน่าเป็นห่วงหรือไม่
```

[ภาพหน้าจอ Workflow ของ AI Agent #1]

AI Agent #2: ผู้ช่วยตอบคำถามจากข้อมูล Real-time

Workflow นี้จะรอรับคำถามจากผู้ใช้ผ่าน Webhook แล้วค้นหาข้อมูลล่าสุดมาให้ AI ช่วยตอบ

โครงสร้าง Workflow: `Webhook` -> `Postgres (Read)` -> `Ollama`

1. **Webhook Node: สร้าง Test URL เพื่อรอรับคำถาม (เช่น `http://localhost:5678/webhook-test/ask-my-agent?question=ตอนนี้ร้อนไหม`)**
2. **Postgres Node:**
    - **Query:**

None

```
SELECT * FROM sensor_data
ORDER BY timestamp DESC
LIMIT 5;
```

    - **(ดึงข้อมูลล่าสุดมา 5 แถว)**
3. **Ollama Node:**
    - **Prompt:**

None

```
คุณคือผู้ช่วยอัจฉริยะที่ต้องตอบคำถามโดยอิงจากข้อมูลจริงเท่านั้น นี่คือข้อมูลล่าสุด 5 รายการจากเซ็นเซอร์:

{{ JSON.stringify($input.all()) }}

จากข้อมูลนี้ โปรดตอบคำถามต่อไปนี้ให้กระชับและตรงไปตรงมาที่สุด: "{{ $json.query.question }}"
```

การทดสอบ: เปิดเบราว์เซอร์แล้วไปที่ URL ของ Webhook พร้อมคำถามของคุณ แล้วกลับมาดู

ผลลัพธ์ใน n8n

ขอแสดงความยินดีด้วย! คุณได้เดินทางมาถึงจุดสิ้นสุดของคู่มือนี้แล้ว ตอนนี้คุณมีระบบ IoT อัจฉริยะที่ครบวงจร ตั้งแต่การเก็บข้อมูลจากฮาร์ดแวร์ไปจนถึงการวิเคราะห์และตอบสนองด้วย AI ซึ่งเป็นพื้นฐานที่แข็งแกร่งให้คุณสามารถนำไปต่อยอดสร้างโปรเจกต์ที่น่าทึ่งอื่นๆ ได้อีกมากมายในอนาคต

ภาคผนวก (Appendix)

ภาคผนวกนี้รวบรวมข้อมูลอ้างอิงที่สำคัญ, แนวทางการแก้ไขปัญหาที่พบบ่อย, และแหล่งข้อมูลเพิ่มเติมที่จะช่วยให้คุณต่อยอดโปรเจกต์ได้อย่างราบรื่น

A: Pinout และข้อมูลทางเทคนิคของบอร์ด SH-BC3

ส่วนนี้จะแสดงแผนผังขา (Pinout) และรายละเอียดทางเทคนิคของบอร์ด Senses-Hub SH-BC3 เพื่อใช้ในการอ้างอิงเมื่อต้องการเชื่อมต่ออุปกรณ์เพิ่มเติม

แผนผังขา (Pinout Diagram):

ข้อมูลทางเทคนิค:

- **Microcontroller: ESP32-C3 (Single-core 32-bit RISC-V, up to 160 MHz)**
- **Connectivity: Wi-Fi 802.11b/g/n, Bluetooth 5 (LE)**
- **Onboard Sensors:**
  - **BME280: I2C Address `0x76`**
  - **GY-521 (MPU-6050): I2C Address `0x68`**
- **Onboard Display:**
  - **OLED SSD1306 (128x64): I2C Address `0x3C`**
- **Operating Voltage: 3.3V**
- **Input Voltage (via USB-C): 5V**

B: การแก้ไขปัญหาที่พบบ่อย (Troubleshooting)

ปัญหาเกี่ยวกับฮาร์ดแวร์ / Arduino:

- **ปัญหา: อัปโหลดโค้ดไม่เข้า / Error: `A fatal error occurred: Failed to connect to ESP32-C3...`**
  - **วิธีแก้: กดปุ่ม BOOT (หรือปุ่มที่มีสัญลักษณ์ `IO9`) บนบอร์ดค้างไว้ แล้วกดปุ่ม RESET หนึ่งครั้ง จากนั้นปล่อยปุ่ม BOOT แล้วลองอัปโหลดอีกครั้ง นี่คือการเข้าสู่ Download Mode ด้วยตนเอง**
- **ปัญหา: คอมพิวเตอร์มองไม่เห็น Port ของบอร์ด**
  - **วิธีแก้:**
    1. **ตรวจสอบว่าสาย USB ที่ใช้เป็นสาย Data (ไม่ใช่สายชาร์จอย่างเดียว)**
    2. **ติดตั้งไดรเวอร์ USB to UART (โดยทั่วไปคือ CH340 หรือ CP210x)**
- **ปัญหา: Serial Monitor แสดงข้อความอ่านไม่ออก (ตัวอักษรต่างดาว)**
  - **วิธีแก้: ตรวจสอบว่าค่า Baud Rate ที่มุมขวาล่างของ Serial Monitor ตรงกับค่าที่ตั้งไว้ในโค้ด (`Serial.begin(115200);`)**

ปัญหาเกี่ยวกับ n8n / PostgreSQL:

- **ปัญหา: n8n ไม่สามารถเชื่อมต่อกับ PostgreSQL ได้ (Credential error)**
  - **วิธีแก้:**
    1. **ตรวจสอบว่า Docker Container ของ PostgreSQL (`my-postgres`) กำลังทำงานอยู่ โดยใช้คำสั่ง `docker ps`**
    2. **ตรวจสอบว่า Host, User, Password, และ Port ใน Credential ของ n8n ถูกต้องตรงกับที่ตั้งค่าไว้ตอนรัน Docker**
- **ปัญหา: n8n ไม่สามารถเชื่อมต่อกับ Local LLM (Ollama/LM Studio) ได้**
  - **วิธีแก้:**
    1. **ตรวจสอบว่าโปรแกรม Ollama หรือ LM Studio กำลังทำงานและได้เปิดใช้งาน API Server แล้ว**

2. ตรวจสอบว่า Base URL ใน Credential ของ Ollama ถูกต้อง (`http://localhost:11434` สำหรับ Ollama, `http://localhost:1234/v1` สำหรับ LM Studio)

C: แหล่งข้อมูลและไลบรารี่เพิ่มเติม

ขั้นตอนการสมัครและใช้งาน Sensesiot.net

1. ไปที่เว็บไซต์: เปิดเว็บเบราว์เซอร์แล้วไปที่ sensesiot.net
2. สมัครสมาชิก: คลิกที่ปุ่ม "Sign Up" และกรอกข้อมูลเพื่อสร้างบัญชีผู้ใช้ใหม่
3. สร้าง Device ใหม่:
    - หลังจากเข้าสู่ระบบแล้ว ให้ไปที่หน้า Dashboard แล้วคลิก "Add Device"
    - ตั้งชื่อ Device ของคุณ (เช่น `My Weather Station`)
    - กำหนดจำนวน Slot ที่ต้องการใช้งาน ในโปรเจกต์ของเราจะใช้ 3 slots ให้ตั้งชื่อแต่ละ Slot ดังนี้:
        - `Slot 1`: Temperature
        - `Slot 2`: Humidity
        - `Slot 3`: Vibration
    - กด "Create Device"
4. คัดลอก Device Key:
    - เมื่อสร้าง Device สำเร็จ ระบบจะแสดงหน้าข้อมูลของ Device นั้นๆ
    - มองหา Device Key ซึ่งเป็นชุดตัวอักษรและตัวเลขยาวๆ
    - คัดลอก Key นี้ เพราะคือ "กุญแจ" ที่บอร์ดของเราจะใช้ในการส่งข้อมูลเข้ามาที่นี่

[ภาพหน้าจอขั้นตอนการสร้าง Device และคัดลอก Key ใน Sensesiot.net]

ลิงก์เอกสารอ้างอิง:

- n8n Documentation: docs.n8n.io - แหล่งข้อมูลที่ดีที่สุดสำหรับการใช้งาน Node ต่างๆ
- Ollama: ollama.com - สำหรับดาวน์โหลดและดูรายการโมเดลที่รองรับ
- LM Studio: lmstudio.ai - สำหรับดาวน์โหลดและเรียนรู้การใช้งาน
- PostgreSQL on Docker: hub.docker.com/_/postgres - คำสั่งและวิธีตั้งค่าเพิ่มเติม
- https://github.com/thomas08/senseshub