|  |  |  |
| --- | --- | --- |
|  | **แผนการจัดการเรียนรู้** | หน่วยที่ 4 |
| รหัสวิชา 21901-2007 วิชา เทคโนโลยีระบบสมองกลฝังตัวและไอโอที | สอนครั้งที่ 13 - 17 |
| ชื่อหน่วยการเรียนรู้ การเชื่อมต่อ IoT Platform (Node-RED & Dashboard) | ทฤษฎี 4 ชม.  ปฏิบัติ 16 ชม. |
| ชื่อเรื่อง/งาน พัฒนาระบบ IoT ด้วย Node-RED และ Dashboard พร้อมจรรยาบรรณการใช้งาน AI | |

**1. ผลลัพธ์การเรียนรู้ระดับหน่วยการเรียน**

1.1 สามารถอธิบายแนวคิดและสถาปัตยกรรมของ IoT (Internet of Things) ได้

1.2 สามารถติดตั้งและใช้งาน Node-RED สำหรับพัฒนาระบบ IoT ได้

1.3 สามารถสร้าง Dashboard สำหรับแสดงผลและควบคุมอุปกรณ์ IoT ได้

1.4 เข้าใจและปฏิบัติตามจรรยาบรรณในการใช้งาน AI และเทคโนโลยีดิจิทัล

**2. อ้างอิงมาตรฐาน/เชื่อมโยงกลุ่มอาชีพ**

**-**

**3. สมรรถนะประจำหน่วย**

3.1 ทักษะการติดตั้งและกำหนดค่า IoT Platform

3.2 ทักษะการออกแบบ User Interface สำหรับ IoT Dashboard

3.3 ทักษะการบูรณาการระบบและแก้ไขปัญหา

3.4 จิตสำนึกด้านจรรยาบรรณและความรับผิดชอบต่อสังคม

**4. จุดประสงค์เชิงพฤติกรรม**

4.1 อธิบายแนวคิด IoT และองค์ประกอบของระบบ IoT ได้

4.2 อธิบายหลักการทำงานของ Node-RED และ Flow-based Programming ได้

4.3 อธิบายหลักการสร้าง Dashboard และ Data Visualization ได้

4.4 ติดตั้งและกำหนดค่า Node-RED บนคอมพิวเตอร์ได้

4.5 สร้าง Dashboard สำหรับแสดงผลข้อมูลและควบคุมอุปกรณ์ได้

4.6 ใช้เครื่องมือ AI อย่างมีจรรยาบรรณในการพัฒนางาน

**5. สาระการเรียนรู้**

5.1 แนวคิดและสถาปัตยกรรมของ IoT (Internet of Things)

5.2 Node-RED และ Flow-based Programming

5.3 โปรโตคอล MQTT (Message Queuing Telemetry Transport)

5.4 การสร้าง Dashboard ด้วย Node-RED

5.5 จรรยาบรรณในการใช้งาน AI และเทคโนโลยีดิจิทัล

**ขั้นสอน (ทฤษฎี 4 ชม.)**

1 แนวคิดและสถาปัตยกรรมของ IoT

กิจกรรมการสอน: บรรยายในชีวิตประจำวัน นักเรียนพบเห็น IoT อะไรบ้าง

เนื้อหา: IoT คืออะไร? ทำไมถึงมีความสำคัญในยุคปัจจุบัน

2 รู้จัก Node-RED และ Flow-based Programming

กิจกรรมการสอน:

* บรรยายและสาธิตการทำงานของ FBP
* สาธิต Flow ง่ายๆ (เช่น Inject -> Debug) เพื่อแสดงแนวคิดการส่ง "Message" (msg.payload)

เนื้อหา:

* Flow-based Programming (FBP) คืออะไร?
* (Nodes, Wires, Flows) Node-RED คืออะไร? ประวัติความเป็นมา และข้อดี (Low-code, Visual)

3 การสร้าง Dashboard และโปรโตคอล MQTT

กิจกรรมการสอน:

* บรรยายเปรียบเทียบข้อดี-ข้อเสีย ระหว่าง Serial (ที่เราจะใช้) กับ MQTT (ที่จะเรียนในอนาคต)
* แสดงตัวอย่าง Dashboard ที่ดีและไม่ดี

เนื้อหา:

* Data Visualization คืออะไร? ทำไมต้องแสดงผลข้อมูล
* ส่วนประกอบของ Dashboard (Gauge, Chart, Switch, Slider)
* แนวคิดของโปรโตคอล MQTT (Publish/Subscribe) เปรียบเทียบกับการสื่อสารแบบ Serial

4 จรรยาบรรณในการใช้งาน AI และเทคโนโลยีดิจิทัล

กิจกรรมการสอน:

* อภิปรายกลุ่มย่อยในหัวข้อ "ถ้าเราใช้ AI เขียนโค้ดทั้งหมด ใครคือเจ้าของผลงาน และใครต้องรับผิดชอบถ้างานผิดพลาด"

เนื้อหา:

* ความรับผิดชอบของนักพัฒนา (Data Privacy, Security)
* ผลกระทบหากระบบ IoT ผิดพลาด (เช่น ในการแพทย์, การควบคุมเครื่องจักร)
* ข้อควรระวัง: AI เป็น "ผู้ช่วย" ไม่ใช่ "ผู้ทำแทน" นักพัฒนาต้องสามารถ "ตรวจสอบ" (Verify) และ "ทำความเข้าใจ" โค้ดหรือ Flow ที่ AI สร้างขึ้นได้

**ขั้นปฏิบัติ (16 ชม.)**

ปฏิบัติการที่ 1: การติดตั้ง Node-RED และสร้าง "Hello, Dashboard"

ปฏิบัติการที่ 2: ใบงานที่ 5 - สร้าง Dashboard ควบคุมฮาร์ดแวร์

**7. สื่อและแหล่งการเรียนรู้**

เอกสารประกอบการสอน เรื่อง "การเชื่อมต่อ IoT Platform (Node-RED & Dashboard)"

สไลด์นำเสนอ

Video สาธิตการต่อวงจรและการเขียนโปรแกรม

ใบงาน/ใบความรู้

**8. หลักฐานการเรียนรู้**

8.1 หลักฐานความรู้

ใบงานทฤษฎี เรื่อง การเชื่อมต่อ IoT Platform (Node-RED & Dashboard)

แบบทดสอบก่อนเรียน-หลังเรียน

สมุดบันทึกการเรียนรู้ (Learning Log)

ใบความรู้ที่มีการจดบันทึกและสรุปประเด็นสำคัญ

8.2 หลักฐานการปฏิบัติงาน

ไฟล์โค้ดโปรแกรม (.ino) ที่สามารถทำงานได้ถูกต้อง

Video/ภาพถ่ายการทำงานตามเงื่อนไขที่กำหนด

**9. การวัดและประเมินผล**

9.1 เกณฑ์การปฏิบัติงาน

ด้านความรู้ (40%)

อธิบายหลักการทำงาน ได้ถูกต้อง ครบถ้วน

ด้านทักษะ (40%)

ต่อวงจรถูกต้องตามหลักการ ไม่เกิดไฟช็อต

เขียนโปรแกรมควบคุมได้อย่างถูกต้อง มีโครงสร้างเป็นระบบ

ด้านเจตคติและกิจนิสัย (20%)

มีความรับผิดชอบต่อการใช้อุปกรณ์และส่งงานตรงเวลา

9.2 วิธีการประเมิน

สังเกตพฤติกรรมการปฏิบัติงานระหว่างเรียน

ประเมินจากชิ้นงานและไฟล์โค้ดโปรแกรม

ประเมินจากรายงานผลการปฏิบัติงาน

9.3 เครื่องมือประเมิน

แบบประเมินทักษะการปฏิบัติงาน

Check List การส่งงาน

แบบประเมินการนำเสนอผลงาน (Presentation Rubric)

**10. บันทึกผลหลังการจัดการเรียนรู้**

10.1 ผลการจัดการเรียนรู้ที่เกิดขึ้นกับผู้เรียน

10.2 ปัญหา อุปสรรคที่พบ

10.3 การแก้ไขปัญหา

|  |  |  |
| --- | --- | --- |
|  | **ใบความรู้ ที่ 4** | หน่วยที่ 4 |
| รหัสวิชา 21901-2007 วิชา เทคโนโลยีระบบสมองกลฝังตัวและไอโอที | สอนครั้งที่ 13 - 17 |
| ชื่อหน่วยการเรียนรู้ การเชื่อมต่อ IoT Platform (Node-RED & Dashboard) | ทฤษฎี 4 ชม.  ปฏิบัติ 16 ชม. |
| ชื่อเรื่อง/งาน พัฒนาระบบ IoT ด้วย Node-RED และ Dashboard พร้อมจรรยาบรรณการใช้งาน AI | |

**1. ผลลัพธ์การเรียนรู้ระดับหน่วยการเรียน**

1.1 สามารถอธิบายแนวคิดและสถาปัตยกรรมของ IoT (Internet of Things) ได้

1.2 สามารถติดตั้งและใช้งาน Node-RED สำหรับพัฒนาระบบ IoT ได้

1.3 สามารถสร้าง Dashboard สำหรับแสดงผลและควบคุมอุปกรณ์ IoT ได้

1.4 เข้าใจและปฏิบัติตามจรรยาบรรณในการใช้งาน AI และเทคโนโลยีดิจิทัล

**2. อ้างอิงมาตรฐาน/เชื่อมโยงกลุ่มอาชีพ**

**-**

**3. สมรรถนะประจำหน่วย**

3.1 ทักษะการติดตั้งและกำหนดค่า IoT Platform

3.2 ทักษะการออกแบบ User Interface สำหรับ IoT Dashboard

3.3 ทักษะการบูรณาการระบบและแก้ไขปัญหา

3.4 จิตสำนึกด้านจรรยาบรรณและความรับผิดชอบต่อสังคม

**4. จุดประสงค์เชิงพฤติกรรม**

4.1 อธิบายแนวคิด IoT และองค์ประกอบของระบบ IoT ได้

4.2 อธิบายหลักการทำงานของ Node-RED และ Flow-based Programming ได้

4.3 อธิบายหลักการสร้าง Dashboard และ Data Visualization ได้

4.4 ติดตั้งและกำหนดค่า Node-RED บนคอมพิวเตอร์ได้

4.5 สร้าง Dashboard สำหรับแสดงผลข้อมูลและควบคุมอุปกรณ์ได้

4.6 ใช้เครื่องมือ AI อย่างมีจรรยาบรรณในการพัฒนางาน

**5. เนื้อหาสาระ**

**5.1 แนวคิดและสถาปัตยกรรมของ IoT**

IoT คืออะไร IoT หรือ Internet of Things... IoT คือแนวคิดที่อุปกรณ์หรือ "สิ่งของ" ต่างๆ (เช่น หลอดไฟ, ตู้เย็น, รถยนต์, หรือแม้แต่เซ็นเซอร์วัดฝุ่น) สามารถเชื่อมต่อและพูดคุยกันได้ผ่านอินเทอร์เน็ต ทำให้มัน "ฉลาด" ขึ้น สามารถทำงานได้เองอัตโนมัติ หรือเราสามารถควบคุมมันจากระยะไกลได้

องค์ประกอบ 4 ส่วนหลักของ IoT ระบบ IoT โดยทั่วไปจะประกอบด้วย 4 ส่วนหลักๆ

1. **Devices/Sensors:** คือตัวฮาร์ดแวร์ที่อยู่หน้างาน (เหมือน ESP32, DHT22, LDR ของเรา) ทำหน้าที่เก็บข้อมูล (เช่น อุณหภูมิ) หรือรอรับคำสั่ง (เช่น รีเลย์)
2. **Gateway:** เปรียบเสมือน "ประตู" ที่รวบรวมข้อมูลจาก Devices หลายๆ ตัว แล้วส่งขึ้นอินเทอร์เน็ต (ในโปรเจกต์ของเรา คอมพิวเตอร์ที่รัน Node-RED และเชื่อมต่อ ESP32 ผ่าน USB ก็ทำหน้าที่คล้ายๆ Gateway)
3. **Cloud/Platform:** คือ "สมอง" ที่อยู่บนอินเทอร์เน็ต ทำหน้าที่เก็บข้อมูลมหาศาล, ประมวลผล, และวิเคราะห์ข้อมูล ใช้ Node-RED เป็น Platform จำลองบนเครื่อง
4. **Application:** คือส่วนที่ผู้ใช้เห็น หรือ "หน้าบ้าน" (เหมือน Dashboard ที่เรากำลังจะสร้าง) ที่ใช้แสดงผลข้อมูลหรือสั่งการควบคุมอุปกรณ์

**5.2 Node-RED และ Flow-based Programming**

Flow-based Programming "Flow-based Programming" หรือ FBP มองการทำงานของโปรแกรมเหมือน "โรงงาน" มี "เครื่องจักร" (เรียกว่า **Nodes**) ที่ทำหน้าที่เฉพาะทางอย่างเดียว เช่น Node รับข้อมูล, Node แปลงข้อมูล, Node ส่งข้อมูล... แล้วเราก็ใช้ "สายพาน" (เรียกว่า **Wires**) เชื่อมต่อเครื่องจักรเหล่านี้เข้าด้วยกัน ข้อมูล (เรียกว่า **Message** หรือ msg) จะไหลไปตามสายพานนี้

Node-RED คืออะไร Node-RED คือเครื่องมือที่เอาแนวคิด FBP มาทำให้เราใช้งานได้ง่ายมากๆ มันเป็น Low-code Platform หมายความว่าเราแทบไม่ต้องเขียนโค้ดเองเลย แต่ใช้การ "ลาก-วาง" Nodes (บล็อก) แล้วลากเส้นเชื่อมต่อกันในหน้าเว็บเบราว์เซอร์แทน

Node-RED จะมีข้อมูลก้อนหนึ่งที่เรียกว่า msg (Message) ซึ่งจะถูกส่งต่อกันไปเรื่อยๆ โดยมีส่วนที่สำคัญที่สุดคือ msg.payload ซึ่งเป็น "กล่อง" ที่ใช้เก็บข้อมูลหลักที่เราสนใจครับ เช่น ถ้าเราส่งเลข 10 ไป msg.payload ก็จะมีค่าเป็น 10

**5.3 โปรโตคอล MQTT**

MQTT ในโปรเจกต์เชื่อมต่อ ESP32 กับคอมพิวเตอร์ด้วยสาย USB (Serial) ซึ่งเป็นการสื่อสารแบบ 1 ต่อ 1... แต่ถ้าเรามีอุปกรณ์ 100 ตัวกระจายอยู่ทั่วโรงเรียน เราคงไม่สามารถลากสาย USB ไปหาทุกตัวได้

นี่คือจุดที่ MQTT (Message Queuing Telemetry Transport) MQTT เป็น "ภาษากลาง" ที่เบามาก เหมาะสำหรับงาน IoT มันทำงานแบบ "Publish/Subscribe" (การส่งและการรับ)

* **Publisher (ผู้ส่ง):** เช่น ESP32 ของเรา ทำหน้าที่ "ส่ง" (Publish) ข้อมูล (เช่น อุณหภูมิ) ไปยัง "หัวข้อ" (Topic) ที่กำหนด (เช่น school/room1/temp)
* **Subscriber (ผู้รับ):** เช่น Node-RED ของเรา ทำหน้าที่ "ติดตาม" (Subscribe) หัวข้อนั้นๆ เมื่อมีข้อมูลใหม่ส่งเข้ามาที่หัวข้อนี้ มันก็จะได้รับข้อมูลทันที
* **Broker (นายหน้า):** เป็นเซิร์ฟเวอร์กลางที่คอยรับข้อมูลจาก Publisher และแจกจ่ายไปยัง Subscriber ที่เกี่ยวข้อง

ข้อดีคือ Publisher และ Subscriber ไม่จำเป็นต้องรู้จักกันเลย คุยกันผ่าน Broker ทำให้ระบบยืดหยุ่นมาก อุปกรณ์จะเพิ่มหรือลดเมื่อไหร่ก็ได้ (ในอนาคต เราจะเปลี่ยนจาก Serial มาใช้ MQTT)

**5.4 การสร้าง Dashboard ด้วย Node-RED (เนื้อหาปฏิบัติการ)**

: ขั้นตอนการทำงานจริง Dashboard "ติดตั้ง" และ "กำหนดค่า" Node-RED ให้รับข้อมูลจาก ESP32 มาแสดงผล และส่งคำสั่งกลับไปควบคุม Relay

**ขั้นที่ 1: การติดตั้ง Node-RED**

1. **ติดตั้ง Node.js:** Node-RED ถูกสร้างด้วย Node.js ดังนั้นเราต้องติดตั้ง Node.js (เวอร์ชัน LTS) ก่อน

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. **ติดตั้ง Node-RED:** เปิด Command Prompt (CMD) แล้วใช้คำสั่ง npm install -g --unsafe-perm node-red คำสั่งนี้จะติดตั้ง Node-RED ลงในเครื่อง

A screenshot of a computer

Description automatically generated

1. **สร้างโฟลเดอร์โปรเจกต์:** ไปที่ Drive D: แล้วสร้างโฟลเดอร์ (เช่น D:\NodeRED\_Project)

A screenshot of a computer program

Description automatically generated

1. **รัน Node-RED:** ใน CMD ย้ายไปที่โฟลเดอร์นั้น (cd D:\NodeRED\_Project) แล้วสั่งรันด้วยคำสั่ง node-red... เมื่อรันแล้ว ให้เราเปิดเบราว์เซอร์ไปที่ http://127.0.0.1:1880/ นี่คือหน้า Editor ของเราครับ

A screenshot of a computer screen

Description automatically generated

**ขั้นที่ 2: ติดตั้ง Nodes ที่จำเป็น (Palette)** (สาธิตการเปิด Manage Palette) Node-RED ไม่ได้มีบล็อกทุกอย่างมาให้ตั้งแต่แรก เราต้องติดตั้งเพิ่มเองครับ

1. ไปที่เมนู (☰) > **Manage palette** > แท็บ **Install**
2. เราต้องติดตั้ง 2 ตัวหลัก:
   * @flowfuse/node-red-dashboard: นี่คือชุดเครื่องมือสำหรับสร้างปุ่ม, เกจวัด, กราฟ (ที่เราเห็นเป็น ui-gauge, ui-switch)

A screenshot of a computer

Description automatically generated

* + node-red-node-serialport: นี่คือบล็อกที่ทำให้ Node-RED คุยผ่านสาย USB (Serial) ได้

A screenshot of a computer

Description automatically generated

**ขั้นที่ 3: ทำความเข้าใจโค้ด ESP32 (ฝั่งฮาร์ดแวร์)** (เปิดโค้ด esp32\_sensor\_relay\_test.ino ให้ดู) โค้ดฝั่ง ESP32 ของเราทำ 2 อย่าง:

1. **ส่งข้อมูล (Part 2):** ทุก 2 วินาที มันจะอ่านค่าเซ็นเซอร์ แล้วสร้างข้อความ JSON (StaticJsonDocument) หน้าตาแบบนี้: {"temp":28.5, "humid":65, "ldr":1500} แล้วส่งผ่าน Serial (Serial.println())
2. **รับคำสั่ง (Part 1):** มันจะคอยเช็คว่ามีข้อความเข้ามาทาง Serial (Serial.available()) หรือไม่ ถ้ามี (เช่น Node-RED ส่ง "R1\_ON" มา) มันก็จะไปสั่ง digitalWrite เพื่อเปิดรีเลย์

**ขั้นที่ 4: สร้าง Flow ใน Node-RED (การลาก-วางบล็อก)** (สาธิตการสร้าง Flow ตามแผนภาพในใบงาน)

**ส่วนที่ 1: การรับข้อมูลมาแสดงผล (Sensor -> Dashboard)**

1. **serial in (บล็อกรับข้อมูล):**
   * ลากบล็อกนี้มาวาง นี่คือ "ประตูทางเข้า"
   * **การตั้งค่า (สำคัญมาก!):** เราต้องดับเบิลคลิกเข้าไปตั้งค่า Port ให้ถูกต้อง กดแว่นขยาย 🔍 เพื่อค้นหา COM Port ของ ESP32 และตั้ง Baud Rate เป็น **115200** ให้ตรงกับในโค้ด Arduino
   * บล็อกนี้จะรอรับข้อมูลจนเจอ \n (ขึ้นบรรทัดใหม่) แล้วส่งข้อมูลทั้งก้อนออกมา
2. **json (บล็อกแปลงข้อมูล):**
   * ลากมาต่อจาก serial in
   * บล็อกนี้ทำหน้าที่ง่ายๆ คือแปลงข้อความ {"temp":...} ที่รับมา ให้กลายเป็น "Object" ที่ Node-RED เข้าใจได้ เพื่อให้บล็อกอื่นดึงข้อมูลไปใช้ง่ายๆ
3. **change (บล็อกแยกข้อมูล):**
   * นี่คือบล็อกที่นักเรียนมักจะสับสนกันครับ... ตอนนี้ msg.payload ของเราเป็นก้อน Object {"temp":..., "humid":...} แต่เกจวัดอุณหภูมิ (Gauge) รับแค่ "ตัวเลข"
   * เราจึงใช้ change มา "ดัก" ข้อมูล
   * **การตั้งค่า (สำคัญมาก!):** เราจะตั้งค่าว่า **Set** msg.payload **to the value** msg.payload.temp
   * ความหมายคือ "ทิ้งข้อมูล msg.payload เก่าที่เป็นก้อนใหญ่ไป แล้วเอาแค่ค่า temp จากข้างใน มาใส่ใน msg.payload ใหม่แทน"
   * เราจะทำแบบนี้ 3 อัน แยกไปสำหรับ temp, humid, และ ldr
4. **ui-gauge / ui-chart (บล็อกแสดงผล):**
   * ลากมาต่อจาก change
   * ดับเบิลคลิกเพื่อตั้งค่า Label (เช่น Temperature), Unit (เช่น °C), และช่วง Min/Max
   * บล็อกเหล่านี้จะรับตัวเลขจาก msg.payload ที่ถูกกรองมาแล้ว ไปแสดงผลบน Dashboard

**ส่วนที่ 2: การส่งคำสั่งไปควบคุม (Dashboard -> Relay)**

1. **ui-switch (บล็อกปุ่มกด):**
   * ลากมาวาง 4 ตัวสำหรับรีเลย์ 4 ช่อง
   * **การตั้งค่า:** เราต้องบอกว่า "ถ้ากดเปิด" (On) ให้ส่ง msg.payload เป็นข้อความ String ว่า R1\_ON... และ "ถ้ากดปิด" (Off) ให้ส่ง R1\_OFF (ต้องตรงกับที่โค้ด ESP32 รอรับเป๊ะๆ)
2. **serial out (บล็อกส่งข้อมูล):**
   * ลากมาวาง 1 ตัว
   * เชื่อม ui-switch ทั้ง 4 ตัวมายัง serial out ตัวนี้
   * **การตั้งค่า:** ต้องตั้งค่า Port และ Baud Rate (115200) ให้เป็น **ตัวเดียวกับ** serial in

**ขั้นที่ 5: การแก้ไขปัญหา (Debugging)** (สาธิตการใช้ Debug Node) เวลาโปรเจกต์ไม่ทำงาน สิ่งที่สำคัญที่สุดคือการ "ไล่ดู" ว่าข้อมูลมันมาถึงไหน และหน้าตาเป็นอย่างไร

* **บล็อก debug (รูปแมลง 🐞):** คือเพื่อนที่ดีที่สุดของเรา
* *ถ้าเกจไม่ขึ้น:* ให้เอา debug ไปต่อหลัง serial in... กด Deploy แล้วดูหน้าต่าง Debug ว่ามีข้อความจาก ESP32 ส่งมาไหม?
* *ถ้ามีข้อความมา:* ให้ย้าย debug ไปต่อหลัง json... ดูว่ามันแปลงเป็น Object ถูกต้องไหม?
* *ถ้าถูกต้อง:* ให้ย้าย debug ไปต่อหลัง change... ดูว่ามันกรองค่าออกมาเหลือแค่ตัวเลขได้ถูกต้องไหม?

**5.5 จรรยาบรรณในการใช้งาน AI และเทคโนโลยีดิจิทัล**

จรรยาบรรณ ในยุคนี้ เรามีเครื่องมือ AI เก่งๆ อย่าง Gemini หรือ ChatGPT ที่ช่วยเราเขียนโค้ด, หาข้อมูล, หรือแม้แต่ช่วยครูสร้างแผนการสอนนี้... มันเป็น "ผู้ช่วย" ที่ดีมาก

แต่ในฐานะนักพัฒนา เรามีสิ่งที่ต้องรับผิดชอบ:

1. **ความปลอดภัย (Security):** ระบบ IoT ของเราอาจถูกแฮกได้ ถ้าเราเขียนโค้ดไม่รัดกุม
2. **ความเป็นส่วนตัว (Privacy):** ข้อมูลเซ็นเซอร์ (เช่น กล้อง, ไมโครโฟน) เป็นข้อมูลส่วนตัว เราต้องจัดการอย่างรับผิดชอบ
3. **ความถูกต้อง (Verification):** AI อาจให้ข้อมูลหรือโค้ดที่ "ผิด" ได้... **หน้าที่ของเราคือต้อง "ตรวจสอบ" และ "ทำความเข้าใจ" ทุกบรรทัดที่ AI สร้างขึ้น** ห้ามคัดลอกมาวางโดยไม่เข้าใจเด็ดขาด
4. **ความรับผิดชอบ (Accountability):** ถ้าโปรเจกต์ที่เราทำ (แม้จะใช้ AI ช่วย) ไปสร้างความเสียหาย... คนที่ต้องรับผิดชอบคือ "เรา" ซึ่งเป็นนักพัฒนา ไม่ใช่ AI

ดังนั้น จงใช้ AI เป็นเครื่องมือเร่งความเร็วในการเรียนรู้ ไม่ใช่ใช้มันแทนที่การเรียนรู้

**6. แบบฝึกหัด/แบบทดสอบ**

**ใบงาน ที่ 6**

**7. เอกสารอ้างอิง**

**-**

**8. ภาคผนวก**

**-**

|  |  |  |
| --- | --- | --- |
|  | **ใบงาน ที่ 6** | หน่วยที่ 4 |
| รหัสวิชา 21901-2007 วิชา เทคโนโลยีระบบสมองกลฝังตัวและไอโอที | สอนครั้งที่ 13 - 17 |
| ชื่อหน่วยการเรียนรู้ การเชื่อมต่อ IoT Platform (Node-RED & Dashboard) | ทฤษฎี 4 ชม.  ปฏิบัติ 16 ชม. |
| ชื่อเรื่อง/งาน พัฒนาระบบ IoT ด้วย Node-RED และ Dashboard พร้อมจรรยาบรรณการใช้งาน AI | |

**1. ผลลัพธ์การเรียนรู้ระดับหน่วยการเรียน**

1.1 สามารถอธิบายแนวคิดและสถาปัตยกรรมของ IoT (Internet of Things) ได้

1.2 สามารถติดตั้งและใช้งาน Node-RED สำหรับพัฒนาระบบ IoT ได้

1.3 สามารถสร้าง Dashboard สำหรับแสดงผลและควบคุมอุปกรณ์ IoT ได้

1.4 เข้าใจและปฏิบัติตามจรรยาบรรณในการใช้งาน AI และเทคโนโลยีดิจิทัล

**2. อ้างอิงมาตรฐาน/เชื่อมโยงกลุ่มอาชีพ**

**-**

**3. สมรรถนะประจำหน่วย**

3.1 ทักษะการติดตั้งและกำหนดค่า IoT Platform

3.2 ทักษะการออกแบบ User Interface สำหรับ IoT Dashboard

3.3 ทักษะการบูรณาการระบบและแก้ไขปัญหา

3.4 จิตสำนึกด้านจรรยาบรรณและความรับผิดชอบต่อสังคม

**4. จุดประสงค์เชิงพฤติกรรม**

4.1 อธิบายแนวคิด IoT และองค์ประกอบของระบบ IoT ได้

4.2 อธิบายหลักการทำงานของ Node-RED และ Flow-based Programming ได้

4.3 อธิบายหลักการสร้าง Dashboard และ Data Visualization ได้

4.4 ติดตั้งและกำหนดค่า Node-RED บนคอมพิวเตอร์ได้

4.5 สร้าง Dashboard สำหรับแสดงผลข้อมูลและควบคุมอุปกรณ์ได้

4.6 ใช้เครื่องมือ AI อย่างมีจรรยาบรรณในการพัฒนางาน

**5. เครื่องมือ วัสดุ และอุปกรณ์**

5.1 Arduino IDE 2..3.6

5.2 ชุดฝึกปฏิบัติ

5.3 Node-red

**6. คำแนะนำ/ข้อควรระวัง**

-

**7. ขั้นตอนการปฏิบัติงาน**

7.1 เตรียมฮาร์ดแวร์และอัปโหลดโค้ด ESP32

7.2 สร้าง Flow ใน Node-RED

7.3 การตั้งค่าและแก้ไขปัญหา

**8. สรุปและวิจารณ์ผล**

-

**9. การประเมินผล**

1. แบบประเมินสมรรถนะงานภาคปฏิบัติ (6 คะแนน)

ความถูกต้องของการต่อวงจร

ความถูกต้องของโปรแกรม

ความสมบูรณ์ของผลลัพธ์

2. แบบสังเกตพฤติกรรมลักษณะนิสัยการทำงาน (4 คะแนน)

ตรงต่อเวลา

ความร่วมมือและวินัย

**10. เอกสารอ้างอิง /เอกสารค้นคว้าเพิ่มเติม**

-