****

**แผนการจัดการเรียนรู้**

หลักสูตรประกาศนียบัตรวิชาชีพ

สาขาวิชาเทคโนโลยีสารสนเทศ

กลุ่มอาชีพซอฟต์แวร์และการประยุกต์

ประเภทวิชาอุตสาหกรรมดิจิทัลและเทคโนโลยีสารสนเทศ

รหัสวิชา 21901-2007 **วิชา** เทคโนโลยีระบบสมองกลฝังตัวและไอโอที

**วิทยาลัยเทคโนโลยีภาคตะวันออก(อี.เทค)**

สำนักงานคณะกรรมการการอาชีวศึกษา กระทรวงศึกษาธิการ

**คำนำ**

รายวิชา 21901-2007 เทคโนโลยีระบบสมองกลฝังตัวและไอโอที เป็นรายวิชาที่มุ่งเน้นการพัฒนาทักษะด้านเทคโนโลยีดิจิทัลที่มีความสำคัญต่อการขับเคลื่อนอุตสาหกรรม 4.0 และการพัฒนาประเทศไทยสู่ Thailand 4.0 โดยเฉพาะในกลุ่มอุตสาหกรรมเป้าหมาย (New S-curve) ที่ต้องการกำลังคนด้านเทคโนโลยีอินเทอร์เน็ตของสรรพสิ่ง (Internet of Things: IoT) และระบบสมองกลฝังตัว ซึ่งเป็นรากฐานสำคัญของการพัฒนาเมืองอัจฉริยะ โรงงานอัจฉริยะ เกษตรอัจฉริยะ และระบบอัตโนมัติต่าง ๆ

แผนการจัดการเรียนรู้รายวิชานี้จัดทำขึ้นตามหลักสูตรประกาศนียบัตรวิชาชีพ พุทธศักราช 2567 ซึ่งเป็นไปตามกรอบคุณวุฒิแห่งชาติ (National Qualifications Framework: NQF) และกรอบคุณวุฒิอ้างอิงอาเซียน (ASEAN Qualifications Reference Framework: AQRF) มาตรฐานการศึกษาของชาติ กรอบคุณวุฒิอาชีวศึกษาแห่งชาติ และเกณฑ์มาตรฐานคุณวุฒิอาชีวศึกษาระดับประกาศนียบัตรวิชาชีพ โดยมุ่งเน้นการเรียนรู้สู่การปฏิบัติเพื่อพัฒนาสมรรถนะของผู้เรียนให้สามารถเขียนโปรแกรมควบคุมอุปกรณ์ IoT จัดเก็บและแสดงผลข้อมูล รวมถึงประยุกต์ใช้ระบบ IoT ในการสร้างโครงงานได้ตามหลักการ พร้อมทั้งพัฒนาคุณธรรม จริยธรรม จรรยาบรรณวิชาชีพ และกิจนิสัยที่ดีในการทำงานด้วยความละเอียดรอบคอบ รับผิดชอบ สื่อสาร คิดเชิงนวัตกรรม และทำงานเป็นทีม

**สารบัญ**

**หน้า**

คำนำ ก

สารบัญ ข

หลักสูตรรายวิชา ข

มาตรฐานอาชีพ ง

ตาราวิเคราะห์หน่วยการเรียนรู้ ฉ

หน่วยการเรียนรู้ ช

ตารางวิเคราะห์พฤติกรรมการเรียนรู้ ฉ

หน่วยที่ 1 เรื่อง/งานพื้นฐาน ESP32 และการเขียนโปรแกรมเบื้องต้น

แผนการจัดการเรียนรู้ 1

ใบความรู้ 4

ใบงานที่ 1 13

หน่วยที่ 2 เรื่อง/งานอุปกรณ์แสดงผลผลลัพธ์ (Display Device)

แผนการจัดการเรียนรู้ 15

ใบความรู้ 19

ใบงานที่ 2 36

ใบงานที่ 3 38

ใบงานที่ 4 40

หน่วยที่ 3 เรื่อง/งานการใช้งาน Sensor และ Actuator

แผนการจัดการเรียนรู้ 42

ใบความรู้ 47

ใบงานที่ 5 59

หน่วยที่ 4 เรื่อง/งานการเชื่อมต่อ IoT Platform (Node-RED & Dashboard)

แผนการจัดการเรียนรู้ 61

ใบความรู้ 64

ใบงาน 71

**หลักสูตรรายวิชา**

**หลักสูตรประกาศนียบัตรวิชาชีพ**

**ประเภทวิชาอุตสาหกรรมดิจิทัลและเทคโนโลยีสารสนเทศ กลุ่มอาชีพ** ซอฟต์แวร์และการประยุกต์

**สาขาวิชา** เทคโนโลยีสารสนเทศ

**รหัส 21901-2007 ชื่อวิชา** เทคโนโลยีระบบสมองกลฝังตัวและไอโอท

**ทฤษฎี 1 ชั่วโมง/สัปดาห์ ปฏิบัติ 3 ชั่วโมง/สัปดาห์ จำนวน 2 หน่วยกิต**

**อ้างอิงมาตรฐาน**

-

**ผลลัพธ์การเรียนรู้ระดับรายวิชา**

เขียนโปรแกรมควบคุมการทำงานอุปกรณ์ไอโอที (Iot) จัดเก็บข้อมูลเพื่อแสดงผลและประยุกต์ใช้

ระบบไอโอที (Iot) ในการสร้างโครงงานตามหลักการ ด้วยความละเอียดรอบคอบ รับผิดชอบ สื่อสาร

คิดเชิงนวัตกรรมและทำงานเป็นทีม

**จุดประสงค์รายวิชา** เพื่อให้

1. มีความรู้เกี่ยวกับสมองกลฝังตัวและการสื่อสารด้วยโพรโทคอลระบบไอโอที (Iot)

2. มีทักษะในการเขียนโปรแกรมควบคุมอุปกรณ์ไอโอที (Iot)

3. มีเจตคติและกิจนิสัยที่ดีในการปฏิบัติงานด้วยความละเอียดรอบคอบ รับผิดชอบ สื่อสาร คิดเชิงนวัตกรรม และทำงานเป็นทีม

4. มีความสามารถประยุกต์ใช้ระบบไอโอที (Iot) ในชีวิตประจำวัน

**สมรรถนะรายวิชา**

1. แสดงความรู้เกี่ยวกับสมองกลฝังตัวและการสื่อสารด้วยโปรโทคอลระบบไอโอที(Iot) ตามหลักการ

2. ใช้งานแพลตฟอร์มไอโอที (Iot) ในการเก็บและแสดงผลลัพธ์การทำงานของระบบไอโอที (Iot)

3. จัดการอุปกรณ์ในระบบไอโอที (Iot) ตามความต้องการในการใช้งาน

4. ประยุกต์ใช้ระบบไอโอที (Iot) ในชีวิตประจำวัน

**คำอธิบายรายวิชา**

ศึกษาและปฏิบัติเกี่ยวกับการโปรแกรมระบบสมองกลฝังตัว สถาปัตยกรรมไอโอที (Iot) พื้นฐานระบบ สื่อสารและโพรโทคอลระบบไอโอที (Iot) การใช้งานแพลตฟอร์มระบบไอโอที ( Iot) เลือกใช้ ไมโครคอนโทรลเลอร์ เทคโนโลยีระบบเซ็นเซอร์ การประยุกต์ใช้งานระบบไอโอที (Iot) เป็นโครงงาน

**ตาราวิเคราะห์หน่วยการเรียนรู้**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ผลลัพธ์การเรียนรู้ระดับรายวิชา(Job)** **: เขียนโปรแกรมควบคุมการทำงานอุปกรณ์ไอโอที (IoT) จัดเก็บข้อมูลเพื่อแสดงผลและประยุกต์ใช้ระบบไอโอที (IoT) ในการสร้างโครงงานตามหลักการ ด้วยความละเอียดรอบคอบ รับผิดชอบ สื่อสาร คิดเชิงนวัตกรรมและทำงานเป็นทีม** | | | | |
| **งานหลัก**  **(Duty)** | **งานย่อย**  **(Task)** | **สมรรถนะย่อย**  (มาตรฐานอาชีพ) | **ความรู้**  **ในการปฏิบัติงาน** | **ทักษะ**  **ในการปฏิบัติงาน** |
| **งานหลัก 1: พื้นฐาน ESP32 และการเขียนโปรแกรมเบื้องต้น** | 1.1 ศึกษาโครงสร้างและหลักการทำงานของ ESP32 | **เขียนโปรแกรม**  **ควบคุมอุปกรณ์ไฟฟ้าเบื้องต้นด้วย ESP32** | **พื้นฐานภาษา C/C++ สำหรับ Embedded System** | **เขียนโปรแกรม**  **ควบคุม GPIO** |
| 1.2 เขียนโปรแกรมควบคุมอุปกรณ์ไฟฟ้าพื้นฐาน |  | **คำสั่งควบคุมการทำงาน (if-else, for, while)** | **ใช้คำสั่งควบคุมการทำงานตามเงื่อนไข** |
| 1.3 สร้างโปรแกรมควบคุมที่ซับซ้อนขึ้น |  | **- การสร้าง Pattern การทำงานของ LED** | **สร้าง Pattern การทำงานที่หลากหลาย** |
| **งานหลัก 2 อุปกรณ์แสดงผลผลลัพธ์ (Display Device)** | 2.1 ศึกษาหลักการทำงานของอุปกรณ์แสดงผล | **แสดงผลข้อมูลจาก ESP32 ด้วยอุปกรณ์แสดงผล** | **โครงสร้างและการทำงานของ 7-Segment Display** | **อ่าน Datasheet และ Pinout** |
| 2.2 เขียนโปรแกรมแสดงผลบน 7-Segment |  | **การแปลงเลขฐาน 10 เป็นรหัส 7-Segment** | **สร้างฟังก์ชันแสดงผลหลายหลัก** |
| 2.3 ใช้งาน LCD I²C |  | **การใช้งาน Library LCD I²C** | **ใช้คำสั่ง lcd.print(), lcd.setCursor()** |
| 2.4 พัฒนาโปรเจกต์แสดงผลข้อมูล |  | **การแสดงผลหลายประเภทร่วมกัน** | **จัดการแสดงผลข้อมูลหลายชนิด** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ผลลัพธ์การเรียนรู้ระดับรายวิชา(Job)** **: เขียนโปรแกรมควบคุมการทำงานอุปกรณ์ไอโอที (IoT) จัดเก็บข้อมูลเพื่อแสดงผลและประยุกต์ใช้ระบบไอโอที (IoT) ในการสร้างโครงงานตามหลักการ ด้วยความละเอียดรอบคอบ รับผิดชอบ สื่อสาร คิดเชิงนวัตกรรมและทำงานเป็นทีม** | | | | |
| **งานหลัก**  **(Duty)** | **งานย่อย**  **(Task)** | **สมรรถนะย่อย**  (มาตรฐานอาชีพ) | **ความรู้**  **ในการปฏิบัติงาน** | **ทักษะ**  **ในการปฏิบัติงาน** |
| **งานหลัก 3: การใช้งาน Sensor และ Actuator** | 3.1 ศึกษาหลักการทำงานของ Sensor | **ใช้งาน Sensor และ Actuator ในการควบคุมระบบอัตโนมัติ** | **ประเภทและหลักการทำงานของ Sensor** | **อ่านค่าจาก Sensor**  **ด้วย Library** |
| 3.2 ศึกษาหลักการทำงานของ Actuator |  | **หลักการทำงานของ Relay Module** | **เขียนโปรแกรม**  **ควบคุม** |
| 1.3 พัฒนาระบบอัตโนมัติ |  | **- Logic การควบคุมตามเงื่อนไข (Condition)** | **สร้าง Logic ควบคุม Actuator ตามเงื่อนไข** |
| **งานหลัก 4: การเชื่อมต่อ IoT Platform (Node-RED & Dashboard)** | 4.1 ศึกษาสถาปัตยกรรม IoT และ Protocol | **เชื่อมต่อและประยุกต์ใช้ ESP32 กับ IoT Platform (Node-RED)** | **สถาปัตยกรรม IoT (Device, Gateway, Cloud)** | **Publish และ Subscribe ข้อมูล** |
| 4.2 ติดตั้งและใช้งาน Node-RED |  | **การติดตั้งและตั้งค่า Node-RED** | **ติดตั้ง Node-RED บนคอมพิวเตอร์** |
| 4.3 สร้าง Dashboard |  | **การใช้งาน Node-RED Dashboard** | **สร้าง Dashboard แสดงข้อมูล Sensor เพิ่ม Widget ควบคุม** |
| 4.4 จริยธรรม AI |  | **การใช้งาน AI เพื่อช่วยอย่างถูกต้อง** | จริยธรรมในการใช้งาน AI |

**ตารางวิเคราะห์พฤติกรรมการเรียนรู้**

**รหัส 21901-2007 ชื่อวิชา** เทคโนโลยีระบบสมองกลฝังตัวและไอโอที

**ทฤษฎี 1 ชั่วโมง/สัปดาห์ ปฏิบัติ 3 ชั่วโมง/สัปดาห์ จำนวน 2 หน่วยกิต**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **หน่วยการเรียนรู้** | **ความสามารถที่คาดหวัง** | | | | | | | | | **รวม** | **จำนวนชั่วโมง**  **ท/ป** |
| **พุทธิพิสัย** | | | | | | **ทักษะ**  **พิสัย** | **จิต**  **พิสัย** | **ประยุกต์**  **ใช้** |
| **ความรู้** | **ความข้าใจ** | **การนำไปใช้** | **การวิเคราะห์** | **การประเมินค่า** | **การสร้างสรรค์** |
| 1.พื้นฐาน ESP32 และการเขียนโปรแกรมเบื้องต้น | **2** | **3** | **3** | **2** | **1** | **2** | **2** | **3** | **2** | **20** | **2/6** |
| 2.อุปกรณ์แสดงผลผลลัพธ์ (Display Device) | **2** | **3** | **4** | **3** | **2** | **3** | **3** | **3** | **3** | **26** | **4/16** |
| 3.การใช้งาน Sensor และ Actuator | **3** | **3** | **4** | **4** | **3** | **4** | **4** | **4** | **4** | **33** | **4/16** |
| 4.การเชื่อมต่อ IoT Platform (Node-RED & Dashboard) | **3** | **4** | **4** | **4** | **4** | **5** | **5** | **4** | **5** | **38** | **4/16** |
| **รวม** | **10** | **13** | **15** | **13** | **10** | **14** | **14** | **13** | **14** |  |  |
| **ประเมินผลลัพธ์การเรียนรู้ระดับรายวิชา** | | | | | | | | | |  | **1/3** |
| **รวมทั้งรายวิชา** | | | | | | | | | |  |  |

**หน่วยการเรียนรู้**

**รหัส 21901-2007 ชื่อวิชา** เทคโนโลยีระบบสมองกลฝังตัวและไอโอที

**ทฤษฎี 1 ชั่วโมง/สัปดาห์ ปฏิบัติ 3 ชั่วโมง/สัปดาห์ จำนวน 2 หน่วยกิต**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **หน่วย**  **ที่** | **หน่วยการเรียนรู้** | **เวลาเรียน (ชม.)** | | |
| **ทฤษฎี** | **ปฏิบัติ** | **รวม** |
| 1 | พื้นฐาน ESP32 และการเขียนโปรแกรมเบื้องต้น | 2 | 6 | 8 |
| 2 | อุปกรณ์แสดงผลผลลัพธ์ (Display Device) | 4 | 16 | 20 |
| 3 | การใช้งาน Sensor และ Actuator | 4 | 16 | 20 |
| 4 | การเชื่อมต่อ IoT Platform (Node-RED & Dashboard) | 4 | 16 | 20 |
|  | **ประเมินผลลัพธ์การเรียนรู้ระดับรายวิชา** | 1 | 3 | 4 |
| **รวม** | | **15** | **57** | **72** |

****

|  |  |  |
| --- | --- | --- |
|  | **แผนการจัดการเรียนรู้** | หน่วยที่1 |
| รหัสวิชา 21901-2007 วิชา เทคโนโลยีระบบสมองกลฝังตัวและไอโอที | สอนครั้งที่ 1 - 2 |
| ชื่อหน่วยการเรียนรู้ พื้นฐาน ESP32 และการเขียนโปรแกรมเบื้องต้น | ทฤษฎี 2 ชม.  ปฏิบัติ 6 ชม. |
| ชื่อเรื่อง/งาน โปรแกรมควบคุม LED 7 หลอด ทำงานตามเงื่อนไข | |

**1. ผลลัพธ์การเรียนรู้ระดับหน่วยการเรียน**

**ผู้เรียนสามารถเขียนโปรแกรมบน ESP32 เพื่อควบคุม LED 7 หลอดให้ทำงานตามเงื่อนไขที่กำหนดได้อย่างถูกต้อง และอธิบายหลักการทำงานของคำสั่งที่ใช้ได้**

**2. อ้างอิงมาตรฐาน/เชื่อมโยงกลุ่มอาชีพ**

**-**

**3. สมรรถนะประจำหน่วย**

3.1 โปรแกรมควบคุม LED 7 หลอด ทำงานตามเงื่อนไข

**4. จุดประสงค์เชิงพฤติกรรม**

4.1 อธิบายโครงสร้างพื้นฐานของบอร์ด ESP32 และหมายเลขพินสำหรับควบคุม LED ได้อย่างถูกต้อง

4.2 เขียนโปรแกรมควบคุม LED 7 หลอดให้ทำงานตามเงื่อนไข (เช่น กะพริบสลับ, วิ่งไล่, แสดงผลตามรูปแบบ) ได้อย่างถูกต้องและมีประสิทธิภาพ

4.3 มีกิจนิสัยการทำงานเป็นระบบ ตรงต่อเวลา และรับผิดชอบต่องานที่ได้รับมอบหมาย

4.4 ปฏิบัติงานด้านการเขียนโปรแกรมและการต่อวงจรตามมาตรฐานความปลอดภัยได้อย่างถูกต้อง

**5. สาระการเรียนรู้**

**5.1 ความรู้พื้นฐานเกี่ยวกับ ESP32 โครงสร้างและส่วนประกอบของบอร์ด ESP32 หมายเลขพิน GPIO และการกำหนดพินสำหรับ Input/Output**

**5.2 การควบคุม LED ด้วย ESP32 หลักการทำงานของ LED และการต่อวงจรพื้นฐาน คำสั่ง pinMode() สำหรับกำหนดโหมดพิน คำสั่ง digitalWrite() สำหรับส่งสัญญาณ HIGH/LOW**

**5.3 โครงสร้างคำสั่งพื้นฐานในภาษา C/C++ คำสั่งควบคุมแบบมีเงื่อนไข (if-else, switch-case) คำสั่งวนซ้ำ (for loop, while loop) ฟังก์ชัน delay() สำหรับหน่วงเวลา**

**5.4 การเขียนโปรแกรมควบคุม LED 7 หลอด การควบคุม LED แบบเดี่ยวและแบบหลายหลอด การสร้างรูปแบบการทำงาน (Pattern)**

**6. กิจกรรมการเรียนรู้**

**ขั้นสอน (ทฤษฎี 2 ชม.)**

**โครงสร้างและส่วนประกอบของ ESP32 พร้อมยกตัวอย่างการใช้งานพิน GPIO อธิบายหลักการทำงานของคำสั่ง pinMode(), digitalWrite() พร้อมตัวอย่างโค้ด อธิบายโครงสร้างคำสั่งควบคุม (if-else, switch-case) และลูป (for, while)**

**ขั้นปฏิบัติ (6 ชม.)**

**กิจกรรมที่ 1: การต่อวงจรและทดสอบ LED**

* **ผู้เรียนต่อวงจร LED 1 หลอดกับ ESP32 ตามแผนผังที่กำหนด**
* **เขียนโปรแกรมควบคุม LED ให้กะพริบเปิด-ปิด**
* **ทดสอบและแก้ไขข้อผิดพลาด**

**กิจกรรมที่ 2: การควบคุม LED 7 หลอดแบบพื้นฐาน**

* **ผู้เรียนต่อวงจร LED 7 หลอดกับบอร์ด ESP32**
* **ทดสอบและปรับปรุงโปรแกรม**

**กิจกรรมที่ 3: การสร้างรูปแบบการทำงานตามเงื่อนไข**

* **ผู้เรียนออกแบบและเขียนโปรแกรมสร้างรูปแบบการทำงาน**
* **ใช้ array และ loop ในการควบคุม LED หลายหลอด**
* **ทดสอบและนำเสนอผลงาน**

**7. สื่อและแหล่งการเรียนรู้**

**เอกสารประกอบการสอน เรื่อง "พื้นฐาน ESP32 และการควบคุม LED"**

**สไลด์นำเสนอ**

**Video สาธิตการต่อวงจรและการเขียนโปรแกรม**

**ใบงาน/ใบความรู้**

**8. หลักฐานการเรียนรู้**

8.1 หลักฐานความรู้

ใบงานทฤษฎี เรื่อง โครงสร้าง ESP32 และคำสั่งพื้นฐาน

แบบทดสอบก่อนเรียน-หลังเรียน

สมุดบันทึกการเรียนรู้ (Learning Log)

ใบความรู้ที่มีการจดบันทึกและสรุปประเด็นสำคัญ

8.2 หลักฐานการปฏิบัติงาน

ไฟล์โค้ดโปรแกรม (.ino) ที่สามารถทำงานได้ถูกต้อง

Video/ภาพถ่ายการทำงานของ LED 7 หลอดตามเงื่อนไขที่กำหนด

**9. การวัดและประเมินผล**

9.1 เกณฑ์การปฏิบัติงาน

ด้านความรู้ (40%)

อธิบายหลักการทำงานของ ESP32 และคำสั่งควบคุม LED ได้ถูกต้อง ครบถ้วน

ด้านทักษะ (40%)

ต่อวงจร LED 7 หลอดได้ถูกต้องตามหลักการ ไม่เกิดไฟช็อต

เขียนโปรแกรมควบคุม LED ได้อย่างถูกต้อง 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

ผู้เรียนส่วนใหญ่เข้าใจหลักการทำงานของคำสั่งพื้นฐานและสามารถประยุกต์ใช้ได้

ผู้เรียนแสดงความสนใจและกระตือรือร้นในการทดลองสร้างรูปแบบการทำงานใหม่ๆ

10.2 ปัญหา อุปสรรคที่พบ

ผู้เรียนบางคนมีพื้นฐานด้านการเขียนโปรแกรมน้อย ทำให้เข้าใจช้ากว่าเพื่อน

เวลาปฏิบัติไม่เพียงพอสำหรับผู้เรียนที่ต้องการทดลองเพิ่มเติม

ผู้เรียนบางคนไม่ส่งรายงานตรงเวลา

10.3 การแก้ไขปัญหา

1) ผลการแก้ไขปัญหาทีส่งผลลัพธ์ที่ดีต่อผู้เรียน

จัดเตรียม Video Tutorial ให้ผู้เรียนสามารถทบทวนได้ด้วยตนเอง

เพิ่มเวลาห้องปฏิบัติการเปิดเพื่อให้ผู้เรียนมาฝึกฝนเพิ่มเติมนอกเวลาเรียน

เน้นย้ำการตรวจสอบวงจรก่อนเปิดไฟทุกครั้ง

2) แนวทางแก้ปัญหาในครั้งต่อไป

พัฒนาสื่อการเรียนรู้ออนไลน์เพิ่มเติม เช่น Interactive Simulation

กำหนดไทม์ไลน์การส่งงานที่ชัดเจนและติดตามอย่างใกล้ชิด

|  |  |  |
| --- | --- | --- |
|  | **ใบความรู้ ที่ 1** | หน่วยที่ 1 |
| รหัสวิชา 21901-2007 วิชา เทคโนโลยีระบบสมองกลฝังตัวและไอโอที | สอนครั้งที่ 1 - 2 |
| ชื่อหน่วยการเรียนรู้ พื้นฐาน ESP32 และการเขียนโปรแกรมเบื้องต้น | ทฤษฎี 2 ชม.  ปฏิบัติ 6 ชม. |
| ชื่อเรื่อง โปรแกรมควบคุม LED 7 หลอด ทำงานตามเงื่อนไข | |

**1. ผลลัพธ์การเรียนรู้ระดับหน่วยการเรียน**

**ผู้เรียนสามารถเขียนโปรแกรมบน ESP32 เพื่อควบคุม LED 7 หลอดให้ทำงานตามเงื่อนไขที่กำหนดได้อย่างถูกต้อง และอธิบายหลักการทำงานของคำสั่งที่ใช้ได้**

**2. อ้างอิงมาตรฐาน/เชื่อมโยงกลุ่มอาชีพ**

**-**

**3. สมรรถนะประจำหน่วย**

3.1 โปรแกรมควบคุม LED 7 หลอด ทำงานตามเงื่อนไข

**4. จุดประสงค์เชิงพฤติกรรม**

4.1 อธิบายโครงสร้างโปรแกรมพื้นฐานใน Arduino IDE ได้

4.2 เขียนโปรแกรมควบคุม LED 7 หลอดให้ทำงานตามเงื่อนไขได้

4.3 ใช้ตัวแปรและโครงสร้างควบคุม (if, for, while) ในการเขียนโปรแกรมได้

4.4 ทดสอบและปรับแก้โค้ดให้ทำงานได้ถูกต้อง

**5. เนื้อหาสาระ**

**ขั้นปฏิบัติ (16 ชม.)**

**1. ESP 32 คืออะไร**

**ESP32 เป็นไมโครคอนโทรลเลอร์ที่พัฒนาโดย Espressif Systems ซึ่งมีฟีเจอร์ที่เหนือกว่าตัว Arduino ทั่วไปในหลายๆ ด้าน เช่น การเชื่อมต่อ Wi-Fi และ Bluetooth ในตัว, ความสามารถในการประมวลผลที่สูงขึ้น และขนาดที่เล็กกว่า**

**2. ประเภทขาใช้งาน**

**- ขาที่เป็นอินพุตอย่างเดียว**

**ขา GPIO34, GPIO35, GPIO36, GPIO39 เป็นอินพุตอย่างเดียว ไม่มีวงจร Pull-up, Pull-down รองรับคำสั่ง analogRead(), digitalRead() เท่านั้น คำสั่งอื่น เช่น digitalWrite(), PWM ไม่รองรับ**

**- Strapping Pins**

**Strapping Pins เป็นขาเกี่ยวข้องกับการทำงานของ ESP32 ในระหว่างการบูตโปรแกรม แบ่งเป็นขาที่ใช้งานแล้วอาจทำให้ ESP32 ไม่สามารถทำงานได้ กับขาที่ใช้งานได้ปกติแต่อาจมีสัญญาณรบกวนออกมาระหว่างบูตโปรแกรม ขาที่ไม่ควรนำมาใช้งานเลย ประกอบด้วยขา GPIO0, GPIO2 ซึ่งเกี่ยวข้องกับการบูตโปรแกรม ขาที่ใช้งานได้ปกติแต่อาจมีสัญญาณรบกวนออกมาระหว่างบูตโปรแกรม ประกอบด้วยขา GPIO12, GPIO15, GPIO5 ไม่ควรนำไปใช้ต่อกับอุปกรณ์ I2C เพราะอาจทำให้เกิดการทำงานผิดปกติของอุปกรณ์ I2C ได้**

**- ขาที่ต่อชิป Flash และ PSRAM**

**ขาที่ต่อชิป Flash เป็นขาที่ห้ามนำมาใช้เด็ดขาดเพราะกระทบกับการอ่านโปรแกรมที่เคยอัพโหลดไว้ หรือทำให้อัพโหลดโปรแกรมไม่เข้า ประกอบด้วยขา GPIO9, GPIO10, GPIO11, GPIO6, GPIO7, GPIO8 ขาที่ต่อชิป PSRAM เป็นขาที่เมื่อนำมาใช้จะไม่สามารถใช้งาน PSRAM ได้ ซึ่ง PSRAM จะมีในบอร์ดที่ระบุอย่างชัดเจนว่ามี PSRAM หากบอร์ดไม่มี PSRAM อยู่แล้วก็ไม่จำเป็นต้องสนใจ แต่หากบอร์ดมี PSRAM ต้องไม่นำขา GPIO16, GPIO17 มาใช้งาน หากนำมาใช้ต้องแน่ใจว่าโปรแกรมที่เขียนไม่เรียกใช้ PSRAM**

**- ขาอัพโหลดโปรแกรม**

**นอกจากขา Strapping Pins ที่เกี่ยวข้องกับการอัพโหลดโปรแกรมและการบูตโปรแกรม ยังมีขา RX, TX ที่ใช้ในการอัพโหลดโปรแกรม หากนำขาดังกล่าวไปใช้ จะทำให้อัพโหลดโปรแกรมไม่ได้ (แต่โปรแกรมที่เคยอัพโหลดไว้ยังทำงานได้ปกติ)**

**- ขา Digital Input**

**ขาที่ใช้อ่านค่าดิจิทัลได้ (ใช้คำสั่ง digitalRead() ได้) คือทุกขาที่ ไม่ใช่ Strapping Pins, ขาที่ต่อชิป Flash และ PSRAM, ขาอัพโหลดโปรแกรม โดยสรุปคือขา GPIO0, GPIO2, GPIO9, GPIO10, GPIO11, GPIO6, GPIO7, GPIO8, RX, TX ใช้ไม่ได้ ที่เหลือใช้ได้หมด**

**- ขา Digital Output**

**ขาที่เขียนค่าดิจิทัลได้ (ใช้คำสั่ง digitalWrite() ได้) คือทุกขาที่ ไม่ใช่ ขาที่เป็นอินพุตอย่างเดียว, Strapping Pins, ขาที่ต่อชิป Flash และ PSRAM, ขาอัพโหลดโปรแกรม โดยสรุปคือขา GPIO0, GPIO2, GPIO9, GPIO10, GPIO11, GPIO6, GPIO7, GPIO8, RX, TX, GPIO34, GPIO35, GPIO36, GPIO39 ใช้ไม่ได้ ที่เหลือใช้ได้หมด**

**- ขา Analog Input**

**ขาที่อ่านค่าแอนะล็อกได้ (ใช้คำสั่ง analogRead() ได้) อ่านค่าได้ 0 ถึง 3.3V แบ่งขาออกเป็น 2 ชุด คือ ADC1 และ ADC2 โดย ADC1 สามารถใช้งานได้เลย ไม่มีเงื่อนไขอะไร ส่วน ADC2 จะใช้งานได้เมื่อปิดใช้ WiFi เท่านั้น ADC1 ใช้งานได้เลย ประกอบด้วยขา GPIO36, GPIO39, GPIO34, GPIO35, GPIO32, GPIO33 ADC2 ต้องปิด WiFi จึงจะใช้งานได้ ประกอบด้วยขา GPIO25, GPIO26, GPIO27, GPIO14, GPIO12, GPIO13, GPIO15, GPIO4**

**- ขา PWM**

**ขาที่อ่านค่าแอนะล็อกได้ (ใช้คำสั่ง analogRead() ได้) อ่านค่าได้ 0 ถึง 3.3V แบ่งขาออกเป็น 2 ชุด คือ ADC1 และ ADC2 โดย ADC1 สามารถใช้งานได้เลย ไม่มีเงื่อนไขอะไร ส่วน ADC2 จะใช้งานได้เมื่อปิดใช้ WiFi เท่านั้น ADC1 ใช้งานได้เลย ประกอบด้วยขา GPIO36, GPIO39, GPIO34, GPIO35, GPIO32, GPIO33 ADC2 ต้องปิด WiFi จึงจะใช้งานได้ ประกอบด้วยขา GPIO25, GPIO26, GPIO27, GPIO14, GPIO12, GPIO13, GPIO15, GPIO4**

**- ขา I2C**

**I2C แบ่งเป็น 2 ชุด คือ I2C0 และ I2C1 โดยปกติใช้เฉพาะ I2C0 ซึ่ง I2C0 ค่าเริ่มต้นกำหนดให้อยู่ที่ขา SDA = GPIO21, SCL = GPIO22 สามารถเปลี่ยนเป็นขาอื่นได้โดยกำหนดในคำสั่ง Wire.begin() ตัวอย่างการย้ายขา SCL ไปที่ GPIO4 และ SDA ไป GPIO5**

**- ขา SPI**

**SPI แบ่งเป็น 2 ชุด คือ HSPI และ VSPI โดยปกติใช้เฉพาะ VSPI ซึ่ง VSPI ค่าเริ่มต้นกำหนดให้อยู่ที่ขา CS = GPIO5, SCK = GPIO18, MISO = GPIO19, MOSI = GPIO23 สามารถเปลี่ยนเป็นขาอื่นได้โดยกำหนดในคำสั่ง SPI.begin() ตัวอย่างการย้ายขา VSPI ไปที่ SCK = GPIO25, MISO = GPIO26, MOSI = GPIO27 และ CS = GPIO15**

**- ขา UART**

**UART แบ่งเป็น 3 ชุด คือ Serial0, Serial1 และ Serial2 โดย Serial0 ใช้อัพโหลดโปรแกรมและสื่อสารกับคอมพิวเตอร์ อยู่ที่ขา TX, RX ไม่ควรนำมาใช้ต่ออุปกรณ์อื่น ส่วน Serial1 และ Serial2 ใช้งานได้อิสระ**

**Serial1 ค่าเริ่มต้นกำหนดให้อยู่ที่ขา RX = GPIO26, TX = GPIO27 เปลี่ยนขาได้โดยใช้คำสั่ง Serial1.setPins()**

**Serial2 ค่าเริ่มต้นกำหนดให้อยู่ที่ขา RX = GPIO4, TX = GPIO25 เปลี่ยนขาได้โดยใช้คำสั่ง Serial2.setPins()**

**ขา TX, RX ที่ย้ายไปใช้ได้ คือทุกขาที่ ไม่ใช่ ขาที่เป็นอินพุตอย่างเดียว, Strapping Pins, ขาที่ต่อชิป Flash และ PSRAM, ขาอัพโหลดโปรแกรม โดยสรุปคือขา GPIO0, GPIO2, GPIO9, GPIO10, GPIO11, GPIO6, GPIO7, GPIO8, RX, TX, GPIO34, GPIO35, GPIO36, GPIO39 ใช้ไม่ได้ ที่เหลือใช้ได้หมด**

**- ขา CAN**

**CAN bus ขา TX, RX ไม่ได้ต่อกับขาใด ๆ เป็นค่าเริ่มต้น ก่อนเริ่มใช้งาน CAN bus ต้องกำหนดขา TX, RX ลงในโค้ดโปรแกรมก่อน โดยขา TX, RX ที่ใช้ได้ คือทุกขาที่ ไม่ใช่ ขาที่เป็นอินพุตอย่างเดียว, Strapping Pins, ขาที่ต่อชิป Flash และ PSRAM, ขาอัพโหลดโปรแกรม โดยสรุปคือขา GPIO0, GPIO2, GPIO9, GPIO10, GPIO11, GPIO6, GPIO7, GPIO8, RX, TX, GPIO34, GPIO35, GPIO36, GPIO39 ใช้ไม่ได้ ที่เหลือใช้ได้หมด**

**- ขา I2S**

**I2S แบ่งเป็น 2 ชุด คือ I2S0 และ I2S1 โดยขา DIN, DOUT, BCLK, WS ไม่ได้ต่อกับขาใด ๆ เป็นค่าเริ่มต้น ก่อนเริ่มใช้งาน I2S ต้องกำหนดขาทั้งหมดลงในโค้ดโปรแกรมก่อน โดยขาที่ใช้ได้ คือทุกขาที่ ไม่ใช่ ขาที่เป็นอินพุตอย่างเดียว, Strapping Pins, ขาที่ต่อชิป Flash และ PSRAM, ขาอัพโหลดโปรแกรม โดยสรุปคือขา GPIO0, GPIO2, GPIO9, GPIO10, GPIO11, GPIO6, GPIO7, GPIO8, RX, TX, GPIO34, GPIO35, GPIO36, GPIO39 ใช้ไม่ได้ ที่เหลือใช้ได้หมด**

| **GPIO** | **Digital Input** | **Digital Output** | **Analog Input** | **PWM** | **I2C / SPI / UART / CAN / I2S** |
| --- | --- | --- | --- | --- | --- |
| **IO36** |  |  |  |  |  |
| **IO39** |  |  |  |  |  |
| **IO34** |  |  |  |  |  |
| **IO35** |  |  |  |  |  |
| **IO32** |  |  |  |  |  |
| **IO33** |  |  |  |  |  |
| **IO25** |  |  |  |  |  |
| **IO26** |  |  |  |  |  |
| **IO27** |  |  |  |  |  |
| **IO14** |  |  |  |  |  |
| **IO12** |  |  |  |  |  |
| **IO13** |  |  |  |  |  |
| **IO9** |  |  |  |  |  |
| **IO10** |  |  |  |  |  |
| **IO11** |  |  |  |  |  |
| **IO6** |  |  |  |  |  |
| **IO7** |  |  |  |  |  |
| **IO8** |  |  |  |  |  |
| **IO15** |  |  |  |  |  |
| **IO2** |  |  |  |  |  |
| **IO0** |  |  |  |  |  |
| **IO4** |  |  |  |  |  |
| **IO16** |  |  |  |  |  |
| **IO17** |  |  |  |  |  |
| **IO5** |  |  |  |  |  |
| **IO18** |  |  |  |  |  |
| **IO19** |  |  |  |  |  |
| **IO21** |  |  |  |  |  |
| **RX** |  |  |  |  |  |
| **TX** |  |  |  |  |  |
| **IO22** |  |  |  |  |  |
| **IO23** |  |  |  |  |  |

3. ติดตั้ง Arduino IDE

• ดาวน์โหลด Arduino IDE จาก Arduino website

• ติดตั้งโปรแกรมโดยทำตามขั้นตอนใน installe

4. ติดตั้ง ESP32 Board

• เปิด Arduino IDE -> ไปที่ "File" -> "Preferences

• เพิ่ม URL: https://dl.espressif.com/dl/package\_esp32\_index.json ใน "Additional Boards Manager URLs

• ไปที่ "Tools" -> "Board" -> "Board Manager"

• ค้นหา "ESP32" และติดตั้งแพ็คเกจ

5. สิ่งที่ควรรู้ก่อนเริ่ม

5.1 โครงสร้างโปรแกรมพื้นฐาน

// ฟังก์ชันเริ่มต้น ทำงานครั้งเดียวตอนเปิดบอร์ดหรือกด Reset

void setup() {

// ใช้สำหรับกำหนดค่าเริ่มต้น เช่น ตั้งค่า pin, เริ่ม Serial

}

// ฟังก์ชันวนซ้ำ ทำงานไปเรื่อย ๆ

void loop() {

// ใช้สำหรับคำสั่งที่ทำซ้ำ เช่น อ่านค่าเซนเซอร์ ควบคุม LED

}

5.2 ตัวแปร (Variables)

5.3 ตัวแปรแบบ Global (ใช้งานได้ทั้งโปรแกรม)

5.4 ตัวแปรแบบ Local (ใช้งานเฉพาะในฟังก์ชัน)

void loop() {

int localVar = 5; // ใช้ได้เฉพาะใน loop()

Serial.println(localVar);

}

5.5 คำสั่งควบคุมขา GPIO

pinMode(pin, mode); // กำหนดโหมดของขา

digitalWrite(pin, value); // เขียนค่า (HIGH / LOW)

digitalRead(pin); // อ่านค่าดิจิทัล (0 / 1)

analogRead(pin); // อ่านค่าอนาล็อก (0–4095 บน ESP32)

analogWrite(pin, value); // เขียนค่า PWM (0–255)

5.6 ฟังก์ชัน (Functions)

// สร้างฟังก์ชัน

void blinkLED(int pin, int timeDelay) {

digitalWrite(pin, HIGH);

delay(timeDelay);

digitalWrite(pin, LOW);

delay(timeDelay);

}

void loop() {

blinkLED(led, 500); // เรียกใช้ฟังก์ชัน

}

พื้นฐานการเรียนเขียนโปรแกรมบควบคุมหลอด LED

****

พื้นฐานการเรียนเขียนโปรแกรมบควบคุมหลอด LED โดยใช้งาน For loop

****

พื้นฐานการเรียนเขียนโปรแกรมบควบคุมหลอด LED โดยใช้งาน For loop สร้างรูปแบบการทำงาน

****

**6. แบบฝึกหัด/แบบทดสอบ**

**ใบงาน ที่ 1**

**7. เอกสารอ้างอิง (ขึ้นหน้าใหม่)**

**-**

**8. ภาคผนวก (เฉลยแบบฝึกหัด เฉลยแบบทดสอบ ฯ)**

**-**

|  |  |  |
| --- | --- | --- |
|  | **ใบงาน ที่ 1** | หน่วยที่ 1 |
| รหัสวิชา 21901-2007 วิชา เทคโนโลยีระบบสมองกลฝังตัวและไอโอที | สอนครั้งที่ 1 - 2 |
| ชื่อหน่วยการเรียนรู้ พื้นฐาน ESP32 และการเขียนโปรแกรมเบื้องต้น | ทฤษฎี 2 ชม.  ปฏิบัติ 6 ชม. |
| ชื่อเรื่อง โปรแกรมควบคุม LED 7 หลอด ทำงานตามเงื่อนไข | |

**1. ผลลัพธ์การเรียนรู้ระดับหน่วยการเรียน**

**ผู้เรียนสามารถเขียนโปรแกรมบน ESP32 เพื่อควบคุม LED 7 หลอดให้ทำงานตามเงื่อนไขที่กำหนดได้อย่างถูกต้อง และอธิบายหลักการทำงานของคำสั่งที่ใช้ได้**

**2. อ้างอิงมาตรฐาน/เชื่อมโยงกลุ่มอาชีพ**

**-**

**3. สมรรถนะประจำหน่วย**

3.1 โปรแกรมควบคุม LED 7 หลอด ทำงานตามเงื่อนไข

**4. จุดประสงค์เชิงพฤติกรรม**

4.1 อธิบายโครงสร้างโปรแกรมพื้นฐานใน Arduino IDE ได้

4.2 เขียนโปรแกรมควบคุม LED 7 หลอดให้ทำงานตามเงื่อนไขได้

4.3 ใช้ตัวแปรและโครงสร้างควบคุม (if, for, while) ในการเขียนโปรแกรมได้

4.4 ทดสอบและปรับแก้โค้ดให้ทำงานได้ถูกต้อง

**5. เครื่องมือ วัสดุ และอุปกรณ์**

5.1 Arduino IDE 2..3.6

5.2 ชุดฝึกปฏิบัติ

**6. คำแนะนำ/ข้อควรระวัง**

-

**7. ขั้นตอนการปฏิบัติงาน**

7.1 ต่อวงจร LED 7 หลอดเข้ากับ ESP32

7.2 เขียนโปรแกรมให้ LED ทำงานตามลำดับที่กำหนด

7.3 ทดสอบและปรับแก้โปรแกรม

**8. สรุปและวิจารณ์ผล**

-

**9. การประเมินผล**

1. แบบประเมินสมรรถนะงานภาคปฏิบัติ (6 คะแนน)

ความถูกต้องของการต่อวงจร

ความถูกต้องของโปรแกรม

ความสมบูรณ์ของผลลัพธ์

2. แบบสังเกตพฤติกรรมลักษณะนิสัยการทำงาน (4 คะแนน)

ตรงต่อเวลา

ความร่วมมือและวินัย

**10. เอกสารอ้างอิง /เอกสารค้นคว้าเพิ่มเติม**

-

|  |  |  |
| --- | --- | --- |
|  | **แผนการจัดการเรียนรู้** | หน่วยที่2 |
| รหัสวิชา 21901-2007 วิชา เทคโนโลยีระบบสมองกลฝังตัวและไอโอที | สอนครั้งที่ 3 - 7 |
| ชื่อหน่วยการเรียนรู้ อุปกรณ์แสดงผลผลลัพธ์ (Display Device) | ทฤษฎี 4 ชม.  ปฏิบัติ 16 ชม. |
| ชื่อเรื่อง/งาน แสดงผลข้อมูลจาก ESP32 ด้วยอุปกรณ์แสดงผล | |

**1. ผลลัพธ์การเรียนรู้ระดับหน่วยการเรียน**

1.1 สามารถอธิบายหลักการทำงานและประเภทของอุปกรณ์แสดงผลได้ถูกต้อง

1.2 สามารถเชื่อมต่อและเขียนโปรแกรมควบคุม 7-Segment Display ทั้งแบบธรรมดาและแบบ I2C ได้

1.3 สามารถเชื่อมต่อและเขียนโปรแกรมแสดงผลบน LCD 16x2 แบบ I2C ได้

1.4 สามารถประยุกต์ใช้สวิตช์ควบคุมการแสดงผลได้อย่างเหมาะสม

1.5 มีทักษะการแก้ไขปัญหาและพัฒนาโปรแกรมอย่างเป็นระบบ

**2. อ้างอิงมาตรฐาน/เชื่อมโยงกลุ่มอาชีพ**

**-**

**3. สมรรถนะประจำหน่วย**

3.1 มีทักษะการแก้ไขปัญหาและพัฒนาโปรแกรมอย่างเป็นระบบ

3.2 ทักษะการเขียนโปรแกรมควบคุมอุปกรณ์แสดงผล

3.3 ทักษะการใช้โปรโตคอลการสื่อสาร I2C

3.4 ทักษะการแก้ไขปัญหาและ Debugging

**4. จุดประสงค์เชิงพฤติกรรม**

4.1 อธิบายหลักการทำงานของ 7-Segment Display แบบธรรมดาและแบบ I2C ได้

4.2 อธิบายโครงสร้างและการทำงานของ LCD 16x2 แบบ I2C ได้

4.3 เชื่อมต่อวงจร 7-Segment Display แบบธรรมดากับ ESP32 ได้ถูกต้อง

4.4 เขียนโปรแกรมแสดงผลตัวเลขบน 7-Segment ได้

4.5 เชื่อมต่อและใช้งาน 7-Segment Display แบบ I2C 4 หลักได้

4.6 เชื่อมต่อและเขียนโปรแกรมแสดงผลบน LCD 16x2 แบบ I2C ได้

4.7 ประยุกต์ใช้สวิตช์ควบคุมการแสดงผลได้

**5. สาระการเรียนรู้**

5.1 อุปกรณ์แสดงผล 7-Segment Display

- โครงสร้างและหลักการทำงานของ 7-Segment

- การต่อวงจรแบบ Common Cathode และ Common Anode

- การแสดงตัวเลข 0-9 และตัวอักษรพื้นฐาน

- การควบคุมด้วย GPIO ของ ESP32**6. กิจกรรมการเรียนรู้**

5.2 การใช้งาน 7-Segment แบบ I2C

- หลักการทำงานของโปรโตคอล I2C

- การเชื่อมต่อ 7-Segment Display 4 หลักแบบ I2C

- ข้อดีของการใช้ I2C (ประหยัด GPIO pins)

- Library สำหรับควบคุม 7-Segment I2C

5.3 LCD 16x2 แบบ I2C

- โครงสร้างและหลักการทำงานของ LCD

- การเชื่อมต่อ LCD แบบ I2C Module

- คำสั่งพื้นฐานในการแสดงผล

- การจัดรูปแบบการแสดงผล

5.4 การรับสัญญาณจากสวิตช์

- การต่อวงจร Pull-up และ Pull-down

- การอ่านค่าสัญญาณดิจิทัลจากสวิตช์

- เทคนิค Debouncing

- การใช้ Interrupt กับสวิตช์

5.5 การประยุกต์ใช้งาน

- ระบบนับขึ้น-ลง

- การควบคุมการกระพริบ

- การจัดการหลายอุปกรณ์แสดงผล

**ขั้นสอน (ทฤษฎี 4 ชม.)**

เนื้อหา: 7-Segment Display แบบธรรมดา

โครงสร้างของ 7-Segment (a-g segments + decimal point) ความแตกต่างระหว่าง Common Cathode และ Common Anode วงจรการต่อใช้งานกับ ESP32 ตารางรหัสการแสดงตัวเลข 0-9 ตัวอย่างโค้ดพื้นฐาน 2C Protocol และ LCD 16x2 โปรโตคอล I2C และการทำงาน 7-Segment I2C และ LCD

กิจกรรมที่ 1: 7-Segment แบบธรรมดาแสดงผลนับตัวเลข

ศึกษาวงจรและต่อวงจร

เขียนโปรแกรมพื้นฐาน

ทดสอบและแก้ไข

กิจกรรมที่ 2: 7-Segment แบบธรรมดาพร้อมสวิตช์ควบคุม

ศึกษาวงจรและต่อวงจร

เขียนโปรแกรมพื้นฐาน

ทดสอบและแก้ไข

กิจกรรมที่ 3: 7-Segment I2C 4 หลัก + สวิตช์ควบคุม

ศึกษาวงจรและต่อวงจร

เขียนโปรแกรมพื้นฐาน

ทดสอบและแก้ไข

กิจกรรมที่ 4: LCD 16x2 I2C แสดงผลข้อมูล

ศึกษาวงจรและต่อวงจร

เขียนโปรแกรมพื้นฐาน

ทดสอบและแก้ไข

**7. สื่อและแหล่งการเรียนรู้**

เอกสารประกอบการสอน เรื่อง "อุปกรณ์แสดงผลผลลัพธ์ (Display Device)"

สไลด์นำเสนอ

Video สาธิตการต่อวงจรและการเขียนโปรแกรม

ใบงาน/ใบความรู้

**8. หลักฐานการเรียนรู้**

8.1 หลักฐานความรู้

ใบงานทฤษฎี เรื่อง แสดงผลข้อมูลจาก ESP32 ด้วยอุปกรณ์แสดงผล

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

ผู้เรียนส่วนใหญ่เข้าใจหลักการทำงานของคำสั่งพื้นฐานและสามารถประยุกต์ใช้ได้

ผู้เรียนแสดงความสนใจและกระตือรือร้นในการทดลองสร้างรูปแบบการทำงานใหม่ๆ

10.2 ปัญหา อุปสรรคที่พบ

ผู้เรียนบางคนมีพื้นฐานด้านการเขียนโปรแกรมน้อย ทำให้เข้าใจช้ากว่าเพื่อน

เวลาปฏิบัติไม่เพียงพอสำหรับผู้เรียนที่ต้องการทดลองเพิ่มเติม

ผู้เรียนบางคนไม่ส่งรายงานตรงเวลา

10.3 การแก้ไขปัญหา

1) ผลการแก้ไขปัญหาทีส่งผลลัพธ์ที่ดีต่อผู้เรียน

จัดเตรียม Video Tutorial ให้ผู้เรียนสามารถทบทวนได้ด้วยตนเอง

เพิ่มเวลาห้องปฏิบัติการเปิดเพื่อให้ผู้เรียนมาฝึกฝนเพิ่มเติมนอกเวลาเรียน

เน้นย้ำการตรวจสอบวงจรก่อนเปิดไฟทุกครั้ง

2) แนวทางแก้ปัญหาในครั้งต่อไป

พัฒนาสื่อการเรียนรู้ออนไลน์เพิ่มเติม เช่น Interactive Simulation

กำหนดไทม์ไลน์การส่งงานที่ชัดเจนและติดตามอย่างใกล้ชิด

|  |  |  |
| --- | --- | --- |
|  | **ใบความรู้ ที่ 2** | หน่วยที่2 |
| รหัสวิชา 21901-2007 วิชา เทคโนโลยีระบบสมองกลฝังตัวและไอโอที | สอนครั้งที่ 3 - 7 |
| ชื่อหน่วยการเรียนรู้ อุปกรณ์แสดงผลผลลัพธ์ (Display Device) | ทฤษฎี 4 ชม.  ปฏิบัติ 16 ชม. |
| ชื่อเรื่อง/งาน แสดงผลข้อมูลจาก ESP32 ด้วยอุปกรณ์แสดงผล | |

**1. ผลลัพธ์การเรียนรู้ระดับหน่วยการเรียน**

1.1 สามารถอธิบายหลักการทำงานและประเภทของอุปกรณ์แสดงผลได้ถูกต้อง

1.2 สามารถเชื่อมต่อและเขียนโปรแกรมควบคุม 7-Segment Display ทั้งแบบธรรมดาและแบบ I2C ได้

1.3 สามารถเชื่อมต่อและเขียนโปรแกรมแสดงผลบน LCD 16x2 แบบ I2C ได้

1.4 สามารถประยุกต์ใช้สวิตช์ควบคุมการแสดงผลได้อย่างเหมาะสม

1.5 มีทักษะการแก้ไขปัญหาและพัฒนาโปรแกรมอย่างเป็นระบบ

**2. อ้างอิงมาตรฐาน/เชื่อมโยงกลุ่มอาชีพ**

**-**

**3. สมรรถนะประจำหน่วย**

3.1 มีทักษะการแก้ไขปัญหาและพัฒนาโปรแกรมอย่างเป็นระบบ

3.2 ทักษะการเขียนโปรแกรมควบคุมอุปกรณ์แสดงผล

3.3 ทักษะการใช้โปรโตคอลการสื่อสาร I2C

3.4 ทักษะการแก้ไขปัญหาและ Debugging

**4. จุดประสงค์เชิงพฤติกรรม**

4.1 อธิบายหลักการทำงานของ 7-Segment Display แบบธรรมดาและแบบ I2C ได้

4.2 อธิบายโครงสร้างและการทำงานของ LCD 16x2 แบบ I2C ได้

4.3 เชื่อมต่อวงจร 7-Segment Display แบบธรรมดากับ ESP32 ได้ถูกต้อง

4.4 เขียนโปรแกรมแสดงผลตัวเลขบน 7-Segment ได้

4.5 เชื่อมต่อและใช้งาน 7-Segment Display แบบ I2C 4 หลักได้

4.6 เชื่อมต่อและเขียนโปรแกรมแสดงผลบน LCD 16x2 แบบ I2C ได้

4.7 ประยุกต์ใช้สวิตช์ควบคุมการแสดงผลได้

**5. เนื้อหาสาระ**

**5.1 อุปกรณ์แสดงผล 7-Segment Display**

**โครงสร้างและหลักการทำงาน**

7-Segment Display เป็นอุปกรณ์แสดงผลตัวเลขที่นิยมใช้กันอย่างแพร่หมายในอุปกรณ์อิเล็กทรอนิกส์ต่างๆ เช่น นาฬิกาดิจิทัล เครื่องชั่งน้ำหนัก เครื่องวัดอุณหภูมิ และอุปกรณ์ที่ต้องการแสดงผลตัวเลข โดย 7-Segment Display ประกอบไปด้วยหลอด LED จำนวน 7 ดวง ที่จัดเรียงในลักษณะรูปตัวเลข 8 โดยแต่ละส่วนของ LED เรียกว่า "Segment" และมีการตั้งชื่อด้วยตัวอักษร a ถึง g ตามตำแหน่งของแต่ละส่วน นอกจากนี้ยังมี decimal point หรือจุดทศนิยมที่อยู่มุมล่างขวาของจอแสดงผลอีกด้วย

การทำงานของ 7-Segment Display อยู่บนหลักการของการเปิด-ปิดหลอด LED แต่ละดวง เมื่อเราต้องการแสดงตัวเลขใดตัวเลขหนึ่ง เราจะควบคุมให้หลอด LED บางดวงติด และบางดวงดับ ตามรูปแบบที่กำหนดไว้ ตัวอย่างเช่น หากต้องการแสดงตัวเลข "0" เราจะต้องเปิดหลอด LED ทุกดวง ยกเว้น segment ตรงกลาง (segment g) หรือหากต้องการแสดงตัวเลข "1" เราจะเปิดเฉพาะ segment b และ c เท่านั้น

7-Segment Display มีอยู่ 2 ประเภทหลักๆ ตามลักษณะการต่อวงจรภายใน คือ แบบ Common Cathode (CC) และแบบ Common Anode (CA) ในแบบ Common Cathode นั้น ขาลบ (Cathode) ของหลอด LED ทุกดวงจะถูกต่อรวมเข้าด้วยกัน และต่อเข้ากับกราวด์ (GND) การควบคุมการเปิด-ปิด LED จะทำได้โดยการส่งสัญญาณ HIGH (3.3V หรือ 5V) ไปยัง segment ที่ต้องการให้ติด และส่งสัญญาณ LOW (0V) ไปยัง segment ที่ต้องการให้ดับ ส่วนแบบ Common Anode นั้นตรงกันข้าม โดยขาบวก (Anode) ของหลอด LED ทุกดวงจะถูกต่อรวมกัน และต่อเข้ากับแหล่งจ่ายไฟบวก การควบคุมจะต้องส่งสัญญาณ LOW เพื่อให้ LED ติด และส่งสัญญาณ HIGH เพื่อให้ LED ดับ

สำหรับการใช้งานกับไมโครคอนโทรลเลอร์อย่าง ESP32 นั้น แบบ Common Cathode จะสะดวกกว่า เพราะ Logic การทำงานเป็นแบบ Active High คือ ส่ง 1 เพื่อเปิด ส่ง 0 เพื่อปิด ซึ่งตรงไปตรงมาและเข้าใจง่ายกว่า อย่างไรก็ตาม ทั้งสองแบบสามารถใช้งานได้ดี เพียงแต่ต้องปรับ Logic ในโปรแกรมให้เหมาะสม

การเลือกใช้ 7-Segment Display ควรพิจารณาถึงขนาดของตัวเลขที่ต้องการแสดง ความสว่างของหลอด LED และกระแสไฟฟ้าที่ใช้งาน โดยทั่วไปแล้ว 7-Segment Display แต่ละ segment จะใช้กระแสประมาณ 10-20 mA ดังนั้นหากเปิด LED ทุกดวง (แสดงตัวเลข 8) จะใช้กระแสรวมประมาณ 140 mA ซึ่งต้องแน่ใจว่าแหล่งจ่ายไฟรองรับได้

**การเชื่อมต่อกับ ESP32**

การเชื่อมต่อ 7-Segment Display เข้ากับ ESP32 จำเป็นต้องใช้ตัวต้านทานจำกัดกระแสสำหรับแต่ละ segment เพื่อป้องกันไม่ให้กระแสไฟฟ้าที่ไหลผ่านหลอด LED มากเกินไปจนทำให้เสียหาย การคำนวณค่าตัวต้านทานที่เหมาะสมใช้สูตร R = (Vcc - Vf) / If โดยที่ Vcc คือแรงดันไฟฟ้าจากขา GPIO ของ ESP32 ซึ่งมีค่า 3.3V, Vf คือแรงดันตกคร่อมหลอด LED (Forward Voltage) ซึ่งโดยทั่วไปมีค่าประมาณ 2.0V, และ If คือกระแสไฟฟ้าที่ต้องการให้ไหลผ่านหลอด LED ซึ่งเราอาจเลือกใช้ 10 mA เพื่อความปลอดภัย

จากการคำนวณ R = (3.3 - 2.0) / 0.01 = 130Ω อย่างไรก็ตาม ในทางปฏิบัติเรามักเลือกใช้ตัวต้านทานขนาด 220Ω ซึ่งเป็นค่ามาตรฐานที่หาง่าย และให้ความปลอดภัยสูงกว่า แม้ว่าหลอด LED จะสว่างน้อยลงเล็กน้อย แต่ก็ยังเพียงพอสำหรับการมองเห็นได้ชัดเจน

ในการต่อวงจร เราจะต้องต่อตัวต้านทาน 220Ω แยกให้กับแต่ละ segment ทั้ง 7 ตัว (a, b, c, d, e, f, g) และอาจรวมถึง decimal point ด้วย แล้วต่อปลายอีกด้านหนึ่งของตัวต้านทานเข้ากับขา GPIO ของ ESP32 ตัวอย่างเช่น segment a ต่อกับ GPIO 12, segment b ต่อกับ GPIO 13, segment c ต่อกับ GPIO 14 และไปเรื่อยๆ ส่วนขา common (COM) ของ 7-Segment Display แบบ Common Cathode จะต่อเข้ากับ GND ของ ESP32

การเลือกขา GPIO นั้นควรหลีกเลี่ยงขาที่มีหน้าที่พิเศษของ ESP32 เช่น GPIO 0, GPIO 2, GPIO 15 ที่ใช้ในการ boot หรือ GPIO 6-11 ที่ต่อกับ Flash Memory ภายใน ควรเลือกใช้ขาที่เป็น General Purpose และไม่มีการใช้งานอื่นๆ เพื่อหลีกเลี่ยงปัญหาในการทำงาน

การต่อวงจรบน Breadboard ควรใช้สายจัมเปอร์ที่มีสีต่างกัน เพื่อให้สามารถตรวจสอบและแก้ไขได้ง่าย อาจใช้สายสีแดงสำหรับสัญญาณ และสายสีดำสำหรับ GND นอกจากนี้ควรตรวจสอบขา Pin ของ 7-Segment Display จาก Datasheet ให้ถูกต้อง เพราะแต่ละรุ่นอาจมีการจัดเรียงขาที่แตกต่างกัน

**การเขียนโปรแกรมควบคุม 7-Segment Display**

การเขียนโปรแกรมควบคุม 7-Segment Display ด้วย ESP32 นั้นเริ่มต้นจากการกำหนดขา GPIO ที่ใช้เชื่อมต่อกับแต่ละ segment ในโปรแกรม เราจะสร้างตัวแปร array เพื่อเก็บหมายเลขขา GPIO เหล่านี้ เช่น int segmentPins[] = {12, 13, 14, 27, 26, 25, 33, 32}; ซึ่งแทนค่า segment a ถึง DP ตามลำดับ จากนั้นในฟังก์ชัน setup() เราจะกำหนดให้ขาเหล่านี้ทำงานเป็น OUTPUT โดยใช้คำสั่ง pinMode(segmentPins[i], OUTPUT); ในลูป

ขั้นตอนสำคัญคือการสร้างตารางรหัสสำหรับแสดงตัวเลข 0-9 โดยเราจะใช้ข้อมูลแบบ binary หรือ hexadecimal เพื่อกำหนดว่าแต่ละตัวเลขจะต้องเปิด-ปิด segment ใดบ้าง ตัวอย่างเช่น ตัวเลข 0 ต้องเปิด segment a, b, c, d, e, f และปิด segment g ซึ่งเราอาจเขียนเป็น binary 0b01111111 หรือ hexadecimal 0x7F สำหรับ Common Cathode

เราจะสร้างฟังก์ชันสำหรับแสดงผลตัวเลข เช่น void displayDigit(int digit) ที่รับค่าตัวเลข 0-9 เป็น parameter แล้วดึงข้อมูลรหัสจากตารางที่เตรียมไว้ จากนั้นวนลูปเขียนค่า HIGH หรือ LOW ไปยังแต่ละขา GPIO ตามรหัสที่กำหนด ตัวอย่างเช่น digitalWrite(segmentPins[i], bitRead(segments[digit], i)); ซึ่งจะอ่านแต่ละ bit จากรหัสและส่งออกไปยัง segment ที่สอดคล้องกัน

ในฟังก์ชัน loop() เราสามารถสร้างการนับตัวเลขจาก 0-9 โดยใช้ตัวแปร counter และเพิ่มค่าทีละ 1 พร้อมเรียกฟังก์ชัน displayDigit(counter) เพื่อแสดงผล จากนั้นใช้ delay(1000) เพื่อหน่วงเวลา 1 วินาที และเมื่อ counter ถึง 9 ก็ให้รีเซ็ตกลับเป็น 0 ด้วยคำสั่ง if (counter > 9) counter = 0;

การเขียนโปรแกรมที่ดีควรแยกโค้ดเป็นฟังก์ชันย่อยๆ เพื่อให้อ่านและดูแลรักษาได้ง่าย เช่น แยกฟังก์ชันการแสดงผล ฟังก์ชันอ่านค่า input และฟังก์ชันควบคุม Logic หลัก นอกจากนี้ควรใส่ comment อธิบายโค้ดเพื่อให้ผู้อื่นหรือตัวเองในอนาคตสามารถเข้าใจได้ง่าย

**5.2 โปรโตคอลการสื่อสาร I2C (Inter-Integrated Circuit)**

**หลักการทำงานของ I2C**

I2C หรือ Inter-Integrated Circuit เป็นโปรโตคอลการสื่อสารแบบอนุกรม (Serial Communication) ที่พัฒนาโดยบริษัท Philips (ปัจจุบันคือ NXP Semiconductors) เพื่อใช้สำหรับการสื่อสารระหว่าง Integrated Circuit ต่างๆ บนแผงวงจรเดียวกัน โดยมีจุดเด่นคือใช้สายสัญญาณเพียง 2 เส้นเท่านั้น แต่สามารถเชื่อมต่ออุปกรณ์หลายตัวเข้าด้วยกันได้ ทำให้ประหยัดขา GPIO ของไมโครคอนโทรลเลอร์ และลดความซับซ้อนของการต่อวงจร

สายสัญญาณทั้ง 2 เส้นของ I2C ประกอบด้วย SCL (Serial Clock Line) ที่ใช้สำหรับสัญญาณนาฬิกาเพื่อซิงโครไนซ์การส่งข้อมูล และ SDA (Serial Data Line) ที่ใช้สำหรับส่งข้อมูลจริง ทั้งสองสายนี้จะต้องมี Pull-up Resistor ต่อขึ้นไปยังแหล่งจ่ายไฟบวก (โดยทั่วไปใช้ 4.7kΩ หรือ 10kΩ) เพื่อให้สัญญาณอยู่ในสถานะ HIGH เมื่อไม่มีการส่งข้อมูล

ในระบบ I2C จะมีอุปกรณ์ 2 ประเภท คือ Master และ Slave โดย Master เป็นผู้ควบคุมการสื่อสาร สร้างสัญญาณนาฬิกา และเริ่มต้น-สิ้นสุดการสื่อสาร ส่วน Slave เป็นอุปกรณ์ที่ถูกควบคุม รอรับคำสั่งจาก Master และตอบสนองตามที่ได้รับคำสั่ง ในระบบหนึ่งสามารถมี Master ได้หลายตัว (Multi-master) และมี Slave ได้มากถึง 128 ตัว (ในทางทฤษฎี) ขึ้นอยู่กับ Address space ที่มี 7 bit

การสื่อสารของ I2C เริ่มต้นจาก Master ส่งสัญญาณ Start Condition โดยการดึง SDA จาก HIGH ลงเป็น LOW ขณะที่ SCL ยังคงเป็น HIGH จากนั้น Master จะส่ง Address ของ Slave ที่ต้องการสื่อสารด้วย 7 bit และตามด้วย bit ที่ 8 เพื่อระบุว่าจะเป็นการ Read หรือ Write (0 = Write, 1 = Read) Slave ที่มี Address ตรงกับที่ถูกเรียกจะตอบกลับด้วยสัญญาณ ACK (Acknowledge) โดยดึง SDA ลงเป็น LOW ในช่วง Clock pulse ถัดไป

หลังจากนั้นจะมีการส่งข้อมูลเป็น byte (8 bit) โดยส่งทีละ bit จาก Most Significant Bit (MSB) ไปยัง Least Significant Bit (LSB) หลังจากส่งครบ 1 byte ผู้รับจะส่ง ACK กลับเพื่อยืนยันว่าได้รับข้อมูลแล้ว กระบวนการนี้จะทำซ้ำไปเรื่อยๆ จนกว่าจะส่งข้อมูลครบตามต้องการ สุดท้าย Master จะส่งสัญญาณ Stop Condition โดยดึง SDA จาก LOW ขึ้นเป็น HIGH ขณะที่ SCL เป็น HIGH เพื่อสิ้นสุดการสื่อสาร

ความเร็วในการสื่อสารของ I2C มีหลายระดับ โดยที่นิยมใช้กันคือ Standard Mode ที่ความเร็ว 100 kHz, Fast Mode ที่ 400 kHz และ Fast Mode Plus ที่ 1 MHz สำหรับ ESP32 นั้นรองรับความเร็วได้หลายระดับ และสามารถกำหนดได้ในโปรแกรม โดยความเร็วที่เหมาะสมขึ้นอยู่กับระยะทางของสายสัญญาณ จำนวนอุปกรณ์ที่ต่อ และความต้องการของแอปพลิเคชัน

**ข้อดีและข้อจำกัดของ I2C**

ข้อดีหลักของ I2C คือความประหยัดในการใช้ขา GPIO เนื่องจากใช้สายเพียง 2 เส้น แต่สามารถเชื่อมต่ออุปกรณ์หลายตัวได้ ตัวอย่างเช่น หากเราต้องการต่อ LCD, RTC, EEPROM และเซนเซอร์อุณหภูมิเข้ากับ ESP32 หากใช้การต่อแบบธรรมดา อาจต้องใช้ GPIO มากกว่า 20 ขา แต่หากใช้ I2C ใช้เพียง 2 ขาเท่านั้น นี่เป็นข้อได้เปรียบอย่างมากสำหรับโปรเจกต์ที่มีการใช้อุปกรณ์หลายชนิด

ข้อดีอีกประการคือโครงสร้างของ I2C ที่ชัดเจนและเป็นมาตรฐาน ทำให้อุปกรณ์จากผู้ผลิตต่างๆ สามารถทำงานร่วมกันได้โดยไม่มีปัญหา นอกจากนี้ยังมี Library สำเร็จรูปมากมายที่รองรับการสื่อสาร I2C ทำให้การพัฒนาโปรแกรมง่ายและรวดเร็วขึ้น นักพัฒนาไม่ต้องกังวลเกี่ยวกับรายละเอียดปลีกย่อยของโปรโตคอล เพียงแค่เรียกใช้ฟังก์ชันที่มีอยู่แล้วในระบบ

อย่างไรก็ตาม I2C ก็มีข้อจำกัดบางประการที่ควรทราบ ข้อแรกคือความเร็วในการสื่อสารที่ต่ำกว่าโปรโตคอลอื่นๆ เช่น SPI ซึ่งอาจไม่เหมาะสำหรับแอปพลิเคชันที่ต้องการถ่ายโอนข้อมูลจำนวนมากในเวลาสั้นๆ เช่น การแสดงผลกราฟิกความละเอียดสูง ข้อจำกัดที่สองคือระยะทางของสายสัญญาณที่จำกัด โดยทั่วไปแนะนำให้ใช้สายไม่เกิน 1 เมตร เพราะหากยาวเกินไปจะเกิดปัญหาเรื่องความจุของสาย (Capacitance) ที่ทำให้สัญญาณบิดเบี้ยว

ปัญหาที่พบบ่อยในการใช้งาน I2C คือการชนกันของ Address เมื่อมีอุปกรณ์หลายตัวที่มี Address เดียวกัน ซึ่งอาจเกิดขึ้นได้หากเราต้องการใช้งานเซนเซอร์ชนิดเดียวกันหลายตัวในระบบเดียว วิธีแก้ไขอาจเป็นการเลือกใช้โมดูลที่มีการปรับ Address ได้ (บางโมดูลมี jumper หรือ solder pad สำหรับเปลี่ยน Address) หรือใช้ I2C Multiplexer เพื่อแยกบัสออกเป็นหลายช่องทาง

การ Debug ปัญหาการสื่อสารของ I2C สามารถทำได้โดยการใช้โปรแกรม I2C Scanner ซึ่งจะสแกนหา Address ของอุปกรณ์ทั้งหมดที่ต่ออยู่บน I2C bus และแสดงผลทาง Serial Monitor ถ้าโปรแกรมไม่พบ Address ของอุปกรณ์ แสดงว่าอาจมีปัญหาเรื่องการต่อวงจร Pull-up resistor หรืออุปกรณ์เสียหาย

**5.3 7-Segment Display แบบ I2C 4 หลัก**

**โครงสร้างและหลักการทำงาน**

7-Segment Display แบบ I2C 4 หลักเป็นโมดูลที่รวม 7-Segment Display จำนวน 4 หลักเข้าด้วยกัน พร้อมวงจรควบคุมแบบ I2C ทำให้สามารถแสดงผลตัวเลข 4 หลัก (0000-9999) ได้โดยใช้สายสัญญาณเพียง 2 เส้น โมดูลที่นิยมใช้กันมากคือแบบที่ใช้ชิป TM1637 ซึ่งเป็น LED Driver IC ที่ออกแบบมาสำหรับควบคุม 7-Segment Display โดยเฉพาะ

TM1637 มีความสามารถในการควบคุม LED ได้สูงสุด 6 หลัก แต่ในโมดูลทั่วไปจะใช้ 4 หลัก เนื่องจากเหมาะสำหรับการแสดงผลตัวเลขทั่วไป เช่น เวลา (12:34), อุณหภูมิ (25.6°C), หรือค่าตัวเลขต่างๆ ชิป TM1637 มีหน่วยความจำภายใน (Display RAM) สำหรับเก็บข้อมูลที่จะแสดงผล ทำให้ไมโครคอนโทรลเลอร์ไม่ต้องส่งข้อมูลซ้ำๆ อยู่ตลอดเวลา เพียงแค่ส่งข้อมูลครั้งเดียวเมื่อต้องการเปลี่ยนค่าที่แสดงผล

การทำงานภายในของ TM1637 ใช้เทคนิคที่เรียกว่า Multiplexing คือการเปิด-ปิดแต่ละหลักของ 7-Segment อย่างรวดเร็วสลับกันไป โดยในแต่ละช่วงเวลาจะมีเพียง 1 หลักเท่านั้นที่ถูกเปิด แต่เนื่องจากความเร็วในการสลับสูงมาก (หลายร้อย Hz) ทำให้ดวงตามนุษย์มองเห็นเป็นการแสดงผลพร้อมกันทุกหลัก เทคนิคนี้ช่วยลดการใช้กระแสไฟฟ้า เพราะไม่ต้องเปิด LED ทุกหลักพร้อมกัน

โมดูล 7-Segment แบบ I2C มักมีขาเชื่อมต่อ 4 ขา คือ VCC (แหล่งจ่ายไฟ 3.3V หรือ 5V), GND (กราวด์), CLK (Clock สำหรับ I2C) และ DIO (Data Input/Output สำหรับ I2C) โมดูลบางรุ่นมี Pull-up resistor ติดมาให้แล้วภายในโมดูล ทำให้ไม่ต้องต่อตัวต้านทานเพิ่มเติม แต่ถ้าต่อระยะไกลหรือมีอุปกรณ์หลายตัว อาจจำเป็นต้องเพิ่ม Pull-up resistor ภายนอก

ข้อดีของการใช้โมดูลนี้คือความสะดวกในการใช้งาน ประหยัดขา GPIO และมี Library สำเร็จรูปให้ใช้งาน เช่น TM1637Display Library ซึ่งมีฟังก์ชันครอบคลุมสำหรับการแสดงผล การควบคุมความสว่าง และการจัดการ colon (เครื่องหมาย : สำหรับแสดงเวลา) นอกจากนี้โมดูลยังมีความเสถียรสูง เพราะวงจรถูกออกแบบมาเป็นพิเศษสำหรับการแสดงผล ไม่ต้องกังวลเรื่องการกระพริบหรือความสว่างไม่สม่ำเสมอ

การเชื่อมต่อโมดูล 7-Segment I2C เข้ากับ ESP32 เป็นเรื่องง่าย เพียงแค่ต่อขา VCC เข้ากับแหล่งจ่ายไฟ 3.3V หรือ 5V (ตรวจสอบจาก Datasheet ของโมดูล) ขา GND เข้ากับ GND ของ ESP32 ขา CLK เข้ากับขา GPIO ที่กำหนดให้เป็น SCL (เช่น GPIO 22) และขา DIO เข้ากับขา GPIO ที่กำหนดให้เป็น SDA (เช่น GPIO 21) ควรใช้สายที่มีคุณภาพดีและไม่ยาวเกินไป เพื่อลดปัญหาสัญญาณรบกวนก่อนเริ่มเขียนโปรแกรม ควรติดตั้ง TM1637Display Library ผ่าน Arduino IDE Library Manager โดยค้นหาคำว่า "TM1637" แล้วเลือกติดตั้ง Library ที่มีชื่อ "TM1637Display by Avishay Orpaz" หรืออาจใช้ Library อื่นๆ ที่รองรับชิป TM1637 ได้เช่นกัน การติดตั้ง Library ทำให้เราสามารถเรียกใช้ฟังก์ชันต่างๆ ได้โดยไม่ต้องเขียนโค้ดควบคุมโปรโตคอลการสื่อสารเองในโปรแกรม เราจะเริ่มต้นด้วยการ include Library และสร้าง object ของ TM1637Display โดยระบุหมายเลขขา CLK และ DIO ที่ใช้ เช่น TM1637Display display(CLK\_PIN, DIO\_PIN); จากนั้นในฟังก์ชัน setup() เราจะเรียกคำสั่ง display.setBrightness(brightness); เพื่อกำหนดความสว่างของหน้าจอ โดยค่า brightness มีช่วงตั้งแต่ 0 (มืดที่สุด) ถึง 7 (สว่างที่สุด) และเรียก display.clear(); เพื่อล้างหน้าจอการแสดงผลตัวเลขสามารถทำได้ง่ายๆ ด้วยคำสั่ง display.showNumberDec(number); โดย number เป็นจำนวนเต็มที่ต้องการแสดง ตัวอย่างเช่น display.showNumberDec(1234); จะแสดงตัวเลข 1234 บนหน้าจอ หากตัวเลขมีหลักน้อยกว่า 4 หลัก ระบบจะเติม leading zero ให้โดยอัตโนมัติ หรือเราสามารถกำหนดให้ไม่แสดง leading zero ด้วยการส่ง parameter เพิ่มเติมสำหรับการแสดงเวลา เช่น 12:34 เราสามารถใช้คำสั่ง display.showNumberDecEx(1234, 0b01000000, true); โดย parameter ตัวที่สองคือ bitmask ที่กำหนดว่าจะเปิด colon (เครื่องหมาย :) หรือไม่ ค่า 0b01000000 หมายถึงเปิด colon ตรงกลาง และ parameter ตัวที่สามคือการกำหนดว่าจะแสดง leading zero หรือไม่ วิธีนี้เหมาะสำหรับการสร้างนาฬิกาดิจิทัลหรือการจับเวลา

**5.4 LCD 16x2 แบบ I2C**

**โครงสร้างและหลักการทำงาน**

LCD (Liquid Crystal Display) 16x2 เป็นจอแสดงผลแบบ Character Display ที่สามารถแสดงผลได้ 16 ตัวอักษร (columns) ใน 2 แถว (rows) รวมทั้งหมด 32 ตัวอักษร LCD ประเภทนี้ใช้หลักการทำงานของ Liquid Crystal ที่สามารถควบคุมให้โปร่งแสงหรือทึบแสงได้โดยการใช้สัญญาณไฟฟ้า เมื่อรวมกับ backlight (แสงพื้นหลัง) ทำให้สามารถมองเห็นตัวอักษรได้ชัดเจน

LCD 16x2 มาตรฐานจะใช้ Controller Chip HD44780 หรือ compatible chips ซึ่งเป็น standard ที่แพร่หมายมาก ชิปนี้มีหน่วยความจำภายใน (DDRAM - Display Data RAM) สำหรับเก็บข้อมูลตัวอักษรที่จะแสดงผล และ CGRAM (Character Generator RAM) สำหรับเก็บ custom characters ที่ผู้ใช้สร้างขึ้นเอง LCD สามารถแสดงผลได้ทั้งตัวอักษร A-Z, a-z, ตัวเลข 0-9, สัญลักษณ์พิเศษต่างๆ และตัวอักษรภาษาญี่ปุ่น (Katakana)

การต่อใช้งาน LCD 16x2 โดยตรงต้องใช้ขา GPIO มากถึง 6-10 ขา (ขึ้นอยู่กับ mode ที่ใช้) ซึ่งค่อนข้างสิ้นเปลือง ดังนั้นจึงมีการพัฒนา I2C Adapter Module ที่แปลงสัญญาณ I2C เป็นสัญญาณควบคุม LCD ทำให้สามารถใช้สายเพียง 2 เส้นในการควบคุม LCD ได้ โมดูล I2C adapter ที่นิยมใช้กันคือแบบที่ใช้ชิป PCF8574 ซึ่งเป็น I2C I/O Expander ที่มี GPIO 8 ขา

PCF8574 จะรับข้อมูลจาก I2C bus แล้วส่งออกเป็นสัญญาณ parallel ไปยังขาต่างๆ ของ LCD โมดูล I2C adapter จะมี potentiometer สำหรับปรับ contrast ของ LCD และ jumper สำหรับเปิด-ปิด backlight ในบางรุ่น นอกจากนี้โมดูลยังมี address jumpers ที่สามารถปรับเปลี่ยน I2C address ได้ โดยทั่วไป default address คือ 0x27 หรือ 0x3F ขึ้นอยู่กับผู้ผลิต

**การเชื่อมต่อและการเขียนโปรแกรม**

การเชื่อมต่อ LCD 16x2 แบบ I2C เข้ากับ ESP32 ทำได้โดยต่อขา VCC เข้ากับ 5V (LCD ส่วนใหญ่ใช้ 5V แม้ว่า ESP32 จะเป็น 3.3V ก็ตาม เพราะโมดูล I2C มี level shifter ในตัว), GND เข้ากับ GND, SDA เข้ากับ GPIO 21 และ SCL เข้ากับ GPIO 22 ของ ESP32 ก่อนใช้งานควรปรับ potentiometer เพื่อให้ตัวอักษรมองเห็นได้ชัดเจน โดยหมุนไปมาจนกว่าจะเห็นตัวอักษรชัดที่สุด

ขั้นตอนแรกในการเขียนโปรแกรมคือการติดตั้ง LiquidCrystal\_I2C Library โดยเปิด Arduino IDE ไปที่ Sketch > Include Library > Manage Libraries แล้วค้นหา "LiquidCrystal I2C" และติดตั้ง Library ที่ชื่อ "LiquidCrystal I2C by Frank de Brabander" หรือ Library อื่นที่รองรับ PCF8574 หากยังไม่แน่ใจว่า LCD ของเรามี address อะไร ให้ใช้โปรแกรม I2C Scanner เพื่อตรวจสอบก่อน

ในโปรแกรม เราจะ include Library และสร้าง object ของ LCD โดยระบุ address, จำนวน columns และ rows เช่น LiquidCrystal\_I2C lcd(0x27, 16, 2); หมายถึง LCD ที่มี address 0x27 แสดงผลได้ 16 ตัวอักษร 2 แถว ในฟังก์ชัน setup() เราจะเรียก lcd.init(); เพื่อเริ่มต้นการทำงานของ LCD และ lcd.backlight(); เพื่อเปิด backlight ทำให้มองเห็นได้ชัดเจนขึ้น

การแสดงผลข้อความทำได้โดยใช้คำสั่ง lcd.print("Hello World"); ซึ่งจะแสดงข้อความที่ตำแหน่ง cursor ปัจจุบัน หากต้องการกำหนดตำแหน่งที่จะแสดงผล ใช้คำสั่ง lcd.setCursor(col, row); โดย col คือหมายเลขคอลัมน์ (0-15) และ row คือหมายเลขแถว (0-1) ตัวอย่างเช่น lcd.setCursor(0, 0); จะย้าย cursor ไปที่มุมซ้ายบนของหน้าจอ

การล้างหน้าจอทำได้ด้วยคำสั่ง lcd.clear(); ซึ่งจะลบข้อความทั้งหมดและย้าย cursor กลับไปที่ตำแหน่ง (0,0) หากต้องการแค่ย้าย cursor กลับไปตำแหน่งเริ่มต้นโดยไม่ลบข้อความ ใช้คำสั่ง lcd.home(); สำหรับการแสดงผลตัวเลข เราสามารถใช้ lcd.print(number); โดยตรง Library จะแปลงตัวเลขเป็นข้อความให้โดยอัตโนมัติ

ฟังก์ชันที่มีประโยชน์อื่นๆ ได้แก่ lcd.cursor(); สำหรับแสดง cursor แบบขีดเส้นใต้, lcd.noCursor(); สำหรับซ่อน cursor, lcd.blink(); สำหรับให้ cursor กระพริบ, lcd.noBlink(); สำหรับหยุดการกระพริบ, lcd.display(); สำหรับเปิดการแสดงผล และ lcd.noDisplay(); สำหรับปิดการแสดงผล (แต่ข้อมูลยังคงอยู่ใน memory)

**การสร้าง Custom Characters**

ความสามารถพิเศษของ LCD 16x2 คือการสร้าง custom characters หรือตัวอักษรพิเศษที่เราออกแบบเอง LCD สามารถเก็บ custom characters ได้สูงสุด 8 ตัว (ตำแหน่ง 0-7) แต่ละตัวอักษรมีขนาด 5x8 pixels เราสามารถออกแบบสัญลักษณ์ต่างๆ เช่น หัวใจ, ลูกศร, กราฟ, ไอคอนพิเศษ หรือตัวอักษรภาษาไทย

การสร้าง custom character เริ่มต้นจากการออกแบบ pixel pattern ซึ่งสามารถใช้กระดาษตารางหรือ online tools เช่น LCD Custom Character Generator จากนั้นแปลง pattern เป็นข้อมูล byte array

ในฟังก์ชัน setup() เราจะเรียกคำสั่ง lcd.createChar(0, heart); เพื่อบันทึก custom character ลงใน CGRAM ที่ตำแหน่ง 0 จากนั้นเมื่อต้องการแสดงผล ใช้คำสั่ง lcd.write(0); เพื่อแสดง custom character ที่ตำแหน่ง 0 เราสามารถสร้าง custom characters หลายตัวและเรียกใช้สลับกันได้ตามต้องการ

การใช้ custom characters ทำให้สามารถสร้าง user interface ที่น่าสนใจขึ้น เช่น แถบ progress bar โดยใช้ custom characters ที่มีความเต็มต่างกันไปในแต่ละตัว, กราฟแท่ง, สัญลักษณ์อุณหภูมิ, หรือไอคอนแบตเตอรี่ ความคิดสร้างสรรค์ในการออกแบบ custom characters สามารถยกระดับความสวยงามและประโยชน์ใช้สอยของ LCD ได้อย่างมาก

**การรับสัญญาณจากสวิตช์ (Push Button)**

**หลักการทำงานของสวิตช์กด**

Push Button Switch หรือสวิตช์กดเป็นอุปกรณ์ input ที่เรียบง่ายแต่สำคัญมาก ใช้สำหรับรับคำสั่งจากผู้ใช้ สวิตช์กดทำงานโดยการเปิด-ปิดวงจรไฟฟ้า เมื่อกดจะทำให้วงจรเชื่อมต่อกัน (Closed) และเมื่อปล่อยจะทำให้วงจรขาดจากกัน (Open) สวิตช์มีหลายประเภท เช่น Momentary Switch ที่เมื่อปล่อยจะกลับสู่สถานะเดิม และ Latching Switch ที่จะค้างอยู่ในสถานะที่กดไว้

สำหรับการใช้งานกับไมโครคอนโทรลเลอร์ เรามักใช้ Momentary Normally Open (NO) Switch ซึ่งในสถานะปกติวงจรจะเปิดอยู่ (ไม่มีการเชื่อมต่อ) และเมื่อกดจึงจะมีการเชื่อมต่อ สวิตช์ประเภทนี้มักมี 4 ขา แต่ที่ใช้งานจริงมีเพียง 2 ขา โดยอีก 2 ขาจะต่อเชื่อมกันอยู่แล้วภายใน สำหรับความแข็งแรงของโครงสร้าง

การต่อสวิตช์เข้ากับไมโครคอนโทรลเลอร์ต้องใช้ตัวต้านทาน Pull-up หรือ Pull-down เพื่อกำหนดสถานะของสัญญาณเมื่อสวิตช์ไม่ได้ถูกกด หากไม่มีตัวต้านทาน สัญญาณจะอยู่ในสถานะ "Floating" ที่ไม่แน่นอน อาจอ่านค่าได้เป็น HIGH หรือ LOW สลับกันไปมาตามสัญญาณรบกวน

**วงจร Pull-down:** ต่อตัวต้านทาน (10kΩ) จากขา GPIO ลงสู่ GND และต่อสวิตช์จากขา GPIO ขึ้นไปยัง VCC เมื่อสวิตช์ไม่ถูกกด สัญญาณจะถูก pull ลงมาเป็น LOW (0V) ผ่านตัวต้านทาน เมื่อกดสวิตช์ สัญญาณจะเป็น HIGH (3.3V) โดยตรงจาก VCC

**วงจร Pull-up:** ต่อตัวต้านทาน (10kΩ) จากขา GPIO ขึ้นไปยัง VCC และต่อสวิตช์จากขา GPIO ลงสู่ GND เมื่อสวิตช์ไม่ถูกกด สัญญาณจะถูก pull ขึ้นเป็น HIGH (3.3V) ผ่านตัวต้านทาน เมื่อกดสวิตช์ สัญญาณจะเป็น LOW (0V) โดยตรงจาก GND

ESP32 มี Internal Pull-up และ Pull-down Resistors อยู่แล้วภายในชิป สามารถเปิดใช้งานได้ผ่านโปรแกรมด้วยคำสั่ง pinMode(pin, INPUT\_PULLUP); หรือ pinMode(pin, INPUT\_PULLDOWN); ทำให้ไม่จำเป็นต้องต่อตัวต้านทานภายนอกในหลายกรณี อย่างไรก็ตาม การใช้ Internal Pull-up เป็นที่นิยมมากกว่า เพราะเสถียรกว่า

**ปัญหา Switch Bouncing และวิธีแก้ไข**

Switch Bouncing เป็นปัญหาที่เกิดขึ้นกับสวิตช์เชิงกลทุกชนิด เมื่อเรากดหรือปล่อยสวิตช์ ส่วนสัมผัสโลหะภายในจะสั่นไหวเล็กน้อย ทำให้เกิดการเปิด-ปิดวงจรหลายครั้งอย่างรวดเร็วในช่วงเวลาสั้นๆ (ประมาณ 10-100 มิลลิวินาที) สำหรับมนุษย์ถือว่าเป็นการกดครั้งเดียว แต่สำหรับไมโครคอนโทรลเลอร์ที่ทำงานเร็วมาก อาจตรวจจับได้เป็นการกดหลายครั้ง

ตัวอย่างปัญหาที่เกิดจาก Switch Bouncing เช่น เราต้องการให้กดสวิตช์แล้ว counter เพิ่มขึ้น 1 ครั้ง แต่เนื่องจาก bouncing ทำให้ counter เพิ่มขึ้น 3-5 ครั้งในการกดเพียงครั้งเดียว หรือในโปรแกรมที่มีการตรวจจับ edge (การเปลี่ยนจาก LOW เป็น HIGH หรือกลับกัน) อาจตรวจจับ edge ได้หลายครั้งในการกดเพียงครั้งเดียว

วิธีแก้ไขปัญหา Bouncing มี 2 แนวทางหลัก คือ Hardware Debouncing และ Software Debouncing

Hardware Debouncing: ใช้ตัวเก็บประจุ (Capacitor) ต่อขนานกับสวิตช์เพื่อทำให้สัญญาณนุ่มนวลขึ้น หรือใช้ IC เฉพาะทาง เช่น 74HC14 Schmitt Trigger แต่วิธีนี้เพิ่มต้นทุนและความซับซ้อนของวงจร จึงไม่ค่อยนิยมในโปรเจกต์ทั่วไป

**Software Debouncing:** เป็นวิธีที่นิยมมากกว่า มีหลายเทคนิค เช่น:

1. **Simple Delay:** หลังจากตรวจจับการเปลี่ยนแปลงสถานะของสวิตช์ ให้หน่วงเวลา 50-100 มิลลิวินาที จากนั้นอ่านค่าสวิตช์อีกครั้งเพื่อยืนยัน ถ้าสถานะยังคงเหมือนเดิม แสดงว่าเป็นการกดจริง

if (digitalRead(buttonPin) == LOW) {

delay(50); *// debounce delay*

if (digitalRead(buttonPin) == LOW) {

*// confirmed button press*

}

}

1. **State Change Detection:** จำสถานะก่อนหน้าไว้ และตรวจจับเฉพาะเมื่อมีการเปลี่ยนสถานะจาก HIGH เป็น LOW หรือกลับกัน พร้อมกับใช้ delay สั้นๆ

int buttonState;

int lastButtonState = HIGH;

void loop() {

buttonState = digitalRead(buttonPin);

if (buttonState != lastButtonState) {

delay(50); *// debounce*

buttonState = digitalRead(buttonPin);

if (buttonState == LOW) {

*// button pressed*

}

}

lastButtonState = buttonState;

}

1. **Millis() Method:** ใช้ฟังก์ชัน millis() ตรวจสอบเวลาที่ผ่านไปนับจากการกดครั้งล่าสุด เพื่อเพิกเฉยการเปลี่ยนแปลงที่เกิดขึ้นเร็วเกินไป วิธีนี้ไม่ทำให้โปรแกรมหยุดทำงานเหมือน delay()

unsigned long lastDebounceTime = 0;

unsigned long debounceDelay = 50;

void loop() {

int reading = digitalRead(buttonPin);

if (reading != lastButtonState) {

lastDebounceTime = millis();

}

if ((millis() - lastDebounceTime) > debounceDelay) {

if (reading != buttonState) {

buttonState = reading;

if (buttonState == LOW) {

*// button pressed*

}

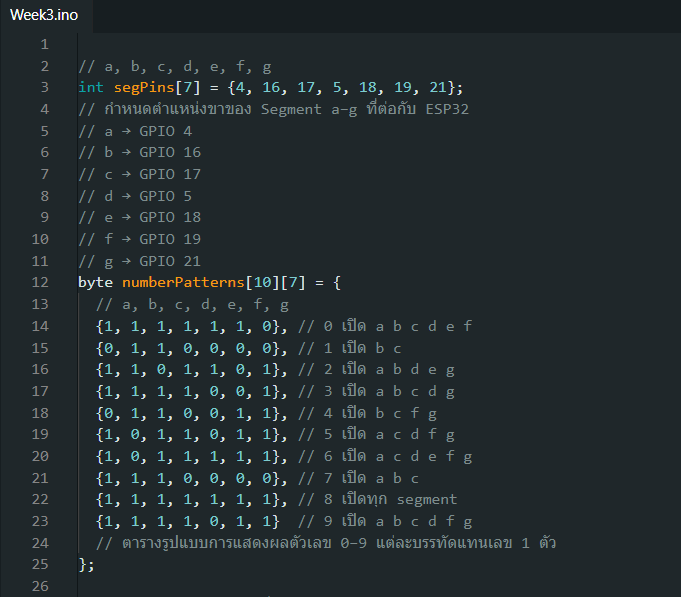
}

}

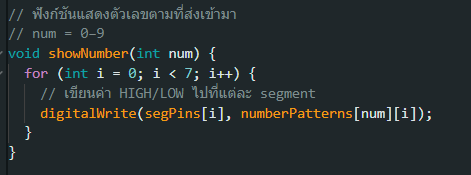
lastButtonState = reading;

}

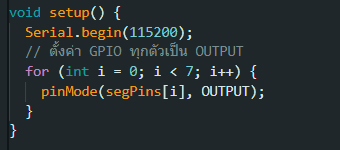
สร้างตารางการแสดงผลชุดตัวเลขบน 7-Segment



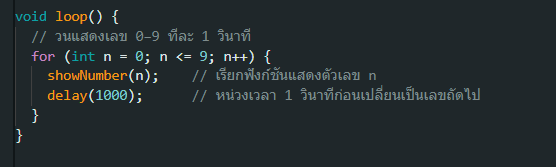
สร้างฟังก์ชันแสดงตัวเลขตามที่ส่งเข้ามา



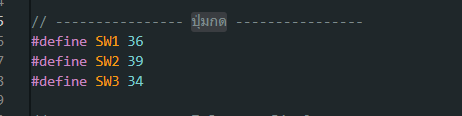
กำหนดรูปแบบโหมดการทำงาน



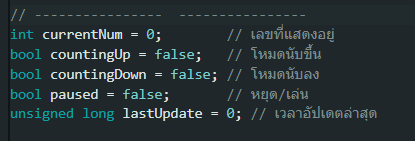
สร้างฟังก์ชันการแสดงผลวนซ้ำ



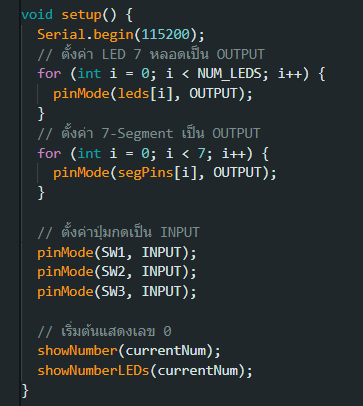
สร้างปุ่มกดเพิ่มรูปแบบการทำงาน



สร้างตัวแปรควบคุมทำงาน



เพิ่มกำหนดรูปแบบโหมดการทำงานของสวิตช์



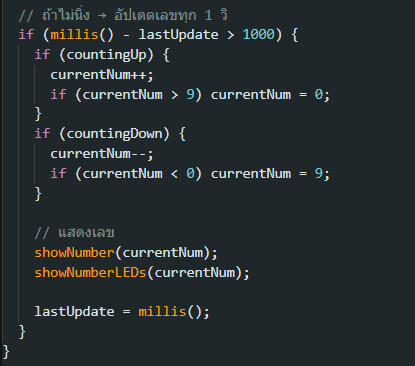
สร้างเงื่อนไขเมื่อมีการกดสวิตช์



สร้างรูปแบบการเล่นแบบกระพริบเมื่อมีการหยุดตัวเลข

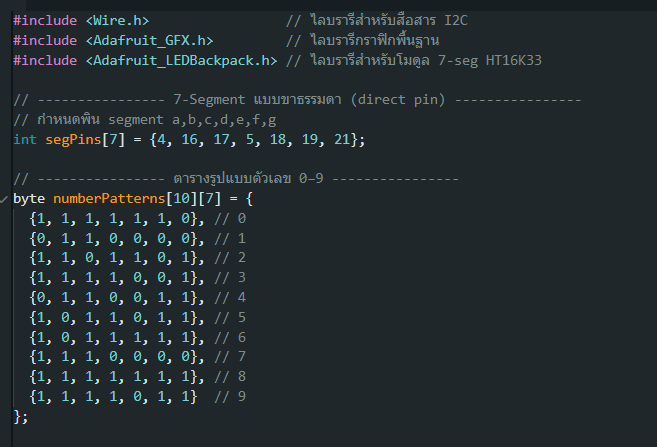


ใช้งานฟังก์ชัน millis() แทน delay

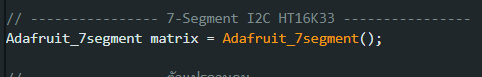


**7-Segment I2C**

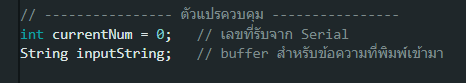
เรียกใช้งานไลบรารีสำหรับสื่อสาร I2C



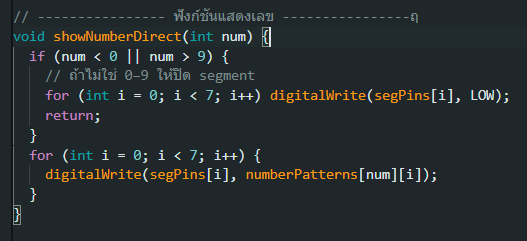
เรียกใช้งาน 7-Segment I2C HT16K33



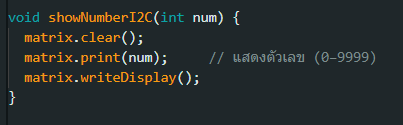
กำหนดตัวแปรควบคุม ตัวแปรควบคุม



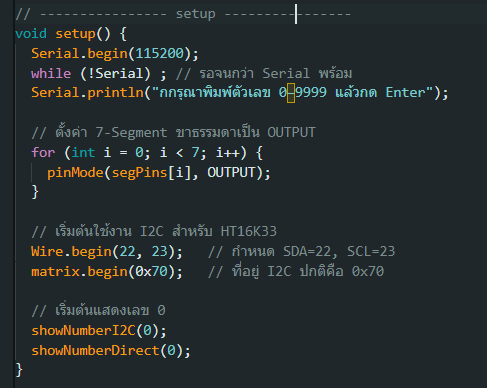
สร้างฟังก์ชันแสดงเลข



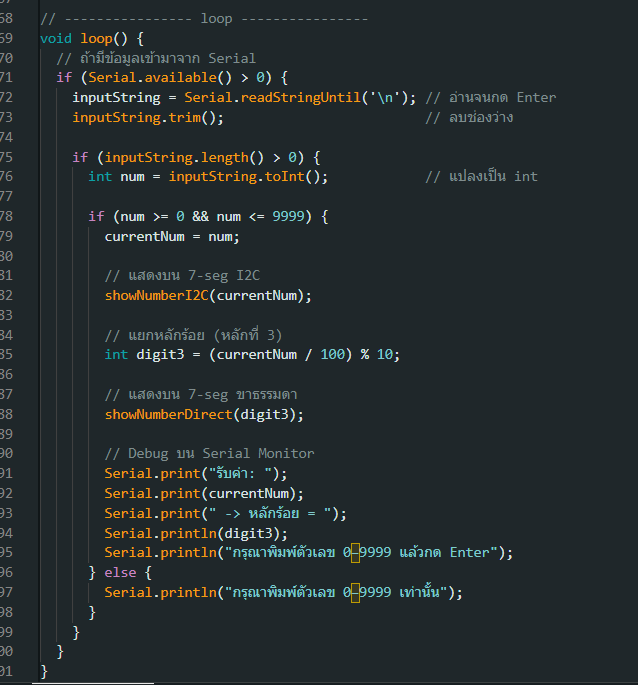
สร้างฟังก์ชัน showNumberI2Cล



สร้างฟังก์ชันกำหนดรูปแบบการทำงานของพินเชื่อมต่อ

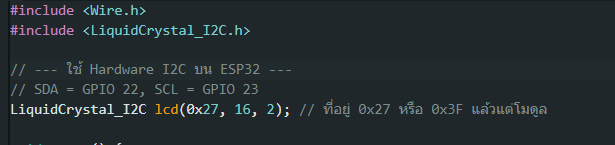


สร้างฟังก์ชันวนซ้ำในการทำงาน

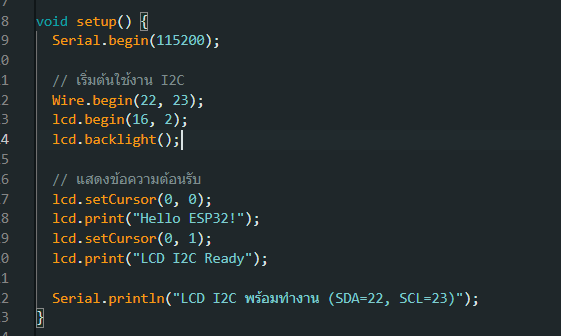


**LCD 16\*2 I2C**

เรียกใช้งานไลบรารีสำหรับสื่อสาร I2C **16\*2 I2C**

****

สร้างฟังก์ชันกำหนดรูปแบบการทำงานของพินเชื่อมต่อ

**.** ****

สร้างฟังก์ชันวนซ้ำในการทำงาน

****

**6. แบบฝึกหัด/แบบทดสอบ**

**ใบงาน ที่ 2 ใบงานที่ 3 และ ใบงานที่ 4**

**7. เอกสารอ้างอิง (ขึ้นหน้าใหม่)**

**-**

**8. ภาคผนวก (เฉลยแบบฝึกหัด เฉลยแบบทดสอบ ฯ)**

**-**

|  |  |  |
| --- | --- | --- |
|  | **ใบงาน ที่ 2** | หน่วยที่2 |
| รหัสวิชา 21901-2007 วิชา เทคโนโลยีระบบสมองกลฝังตัวและไอโอที | สอนครั้งที่ 3 - 7 |
| ชื่อหน่วยการเรียนรู้ อุปกรณ์แสดงผลผลลัพธ์ (Display Device) | ทฤษฎี 4 ชม.  ปฏิบัติ 16 ชม. |
| ชื่อเรื่อง/งาน แสดงผลข้อมูลจาก ESP32 ด้วยอุปกรณ์แสดงผล | |

**1. ผลลัพธ์การเรียนรู้ระดับหน่วยการเรียน**

1.1 สามารถอธิบายหลักการทำงานและประเภทของอุปกรณ์แสดงผลได้ถูกต้อง

1.2 สามารถเชื่อมต่อและเขียนโปรแกรมควบคุม 7-Segment Display ทั้งแบบธรรมดาและแบบ I2C ได้

1.3 สามารถเชื่อมต่อและเขียนโปรแกรมแสดงผลบน LCD 16x2 แบบ I2C ได้

1.4 สามารถประยุกต์ใช้สวิตช์ควบคุมการแสดงผลได้อย่างเหมาะสม

1.5 มีทักษะการแก้ไขปัญหาและพัฒนาโปรแกรมอย่างเป็นระบบ

**2. อ้างอิงมาตรฐาน/เชื่อมโยงกลุ่มอาชีพ**

**-**

**3. สมรรถนะประจำหน่วย**

3.1 มีทักษะการแก้ไขปัญหาและพัฒนาโปรแกรมอย่างเป็นระบบ

3.2 ทักษะการเขียนโปรแกรมควบคุมอุปกรณ์แสดงผล

3.3 ทักษะการใช้โปรโตคอลการสื่อสาร I2C

3.4 ทักษะการแก้ไขปัญหาและ Debugging

**4. จุดประสงค์เชิงพฤติกรรม**

4.1 อธิบายหลักการทำงานของ 7-Segment Display แบบธรรมดาและแบบ I2C ได้

4.2 อธิบายโครงสร้างและการทำงานของ LCD 16x2 แบบ I2C ได้

4.3 เชื่อมต่อวงจร 7-Segment Display แบบธรรมดากับ ESP32 ได้ถูกต้อง

4.4 เขียนโปรแกรมแสดงผลตัวเลขบน 7-Segment ได้

4.5 เชื่อมต่อและใช้งาน 7-Segment Display แบบ I2C 4 หลักได้

4.6 เชื่อมต่อและเขียนโปรแกรมแสดงผลบน LCD 16x2 แบบ I2C ได้

4.7 ประยุกต์ใช้สวิตช์ควบคุมการแสดงผลได้

**5. เครื่องมือ วัสดุ และอุปกรณ์**

5.1 Arduino IDE 2..3.6

5.2 ชุดฝึกปฏิบัติ

**6. คำแนะนำ/ข้อควรระวัง**

-

**7. ขั้นตอนการปฏิบัติงาน**

7.1 ต่อวงจร 7-Segment 1 หลักเข้ากับ ESP32

7.3 เพิ่มสวิตช์ ในการสร้างเงื่อนไขการทำงาน

7.4 เขียนโปรแกรมให้7-Segment 1 หลักทำงานตามลำดับที่กำหนด

7.5 ทดสอบและปรับแก้โปรแกรม

**8. สรุปและวิจารณ์ผล**

-

**9. การประเมินผล**

1. แบบประเมินสมรรถนะงานภาคปฏิบัติ (6 คะแนน)

ความถูกต้องของการต่อวงจร

ความถูกต้องของโปรแกรม

ความสมบูรณ์ของผลลัพธ์

2. แบบสังเกตพฤติกรรมลักษณะนิสัยการทำงาน (4 คะแนน)

ตรงต่อเวลา

ความร่วมมือและวินัย

**10. เอกสารอ้างอิง /เอกสารค้นคว้าเพิ่มเติม**

-

|  |  |  |
| --- | --- | --- |
|  | **ใบงาน ที่ 3** | หน่วยที่2 |
| รหัสวิชา 21901-2007 วิชา เทคโนโลยีระบบสมองกลฝังตัวและไอโอที | สอนครั้งที่ 3 - 7 |
| ชื่อหน่วยการเรียนรู้ อุปกรณ์แสดงผลผลลัพธ์ (Display Device) | ทฤษฎี 4 ชม.  ปฏิบัติ 16 ชม. |
| ชื่อเรื่อง/งาน แสดงผลข้อมูลจาก ESP32 ด้วยอุปกรณ์แสดงผล | |

**1. ผลลัพธ์การเรียนรู้ระดับหน่วยการเรียน**

1.1 สามารถอธิบายหลักการทำงานและประเภทของอุปกรณ์แสดงผลได้ถูกต้อง

1.2 สามารถเชื่อมต่อและเขียนโปรแกรมควบคุม 7-Segment Display ทั้งแบบธรรมดาและแบบ I2C ได้

1.3 สามารถเชื่อมต่อและเขียนโปรแกรมแสดงผลบน LCD 16x2 แบบ I2C ได้

1.4 สามารถประยุกต์ใช้สวิตช์ควบคุมการแสดงผลได้อย่างเหมาะสม

1.5 มีทักษะการแก้ไขปัญหาและพัฒนาโปรแกรมอย่างเป็นระบบ

**2. อ้างอิงมาตรฐาน/เชื่อมโยงกลุ่มอาชีพ**

**-**

**3. สมรรถนะประจำหน่วย**

3.1 มีทักษะการแก้ไขปัญหาและพัฒนาโปรแกรมอย่างเป็นระบบ

3.2 ทักษะการเขียนโปรแกรมควบคุมอุปกรณ์แสดงผล

3.3 ทักษะการใช้โปรโตคอลการสื่อสาร I2C

3.4 ทักษะการแก้ไขปัญหาและ Debugging

**4. จุดประสงค์เชิงพฤติกรรม**

4.1 อธิบายหลักการทำงานของ 7-Segment Display แบบธรรมดาและแบบ I2C ได้

4.2 อธิบายโครงสร้างและการทำงานของ LCD 16x2 แบบ I2C ได้

4.3 เชื่อมต่อวงจร 7-Segment Display แบบธรรมดากับ ESP32 ได้ถูกต้อง

4.4 เขียนโปรแกรมแสดงผลตัวเลขบน 7-Segment ได้

4.5 เชื่อมต่อและใช้งาน 7-Segment Display แบบ I2C 4 หลักได้

4.6 เชื่อมต่อและเขียนโปรแกรมแสดงผลบน LCD 16x2 แบบ I2C ได้

4.7 ประยุกต์ใช้สวิตช์ควบคุมการแสดงผลได้

**5. เครื่องมือ วัสดุ และอุปกรณ์**

5.1 Arduino IDE 2..3.6

5.2 ชุดฝึกปฏิบัติ

**6. คำแนะนำ/ข้อควรระวัง**

-

**7. ขั้นตอนการปฏิบัติงาน**

7.1 ต่อวงจร 7-Segment 4 หลัก รูปแบบ I2C เข้ากับ ESP32

7.3 เพิ่มสวิตช์ ในการสร้างเงื่อนไขการทำงาน

7.4 เขียนโปรแกรมให้ 7-Segment 4 หลัก รูปแบบ I2C ทำงานตามลำดับที่กำหนด

7.5 ทดสอบและปรับแก้โปรแกรม

**8. สรุปและวิจารณ์ผล**

-

**9. การประเมินผล**

1. แบบประเมินสมรรถนะงานภาคปฏิบัติ (6 คะแนน)

ความถูกต้องของการต่อวงจร

ความถูกต้องของโปรแกรม

ความสมบูรณ์ของผลลัพธ์

2. แบบสังเกตพฤติกรรมลักษณะนิสัยการทำงาน (4 คะแนน)

ตรงต่อเวลา

ความร่วมมือและวินัย

**10. เอกสารอ้างอิง /เอกสารค้นคว้าเพิ่มเติม**

-

|  |  |  |
| --- | --- | --- |
|  | **ใบงาน ที่ 4** | หน่วยที่2 |
| รหัสวิชา 21901-2007 วิชา เทคโนโลยีระบบสมองกลฝังตัวและไอโอที | สอนครั้งที่ 3 - 7 |
| ชื่อหน่วยการเรียนรู้ อุปกรณ์แสดงผลผลลัพธ์ (Display Device) | ทฤษฎี 4 ชม.  ปฏิบัติ 16 ชม. |
| ชื่อเรื่อง/งาน แสดงผลข้อมูลจาก ESP32 ด้วยอุปกรณ์แสดงผล | |

**1. ผลลัพธ์การเรียนรู้ระดับหน่วยการเรียน**

1.1 สามารถอธิบายหลักการทำงานและประเภทของอุปกรณ์แสดงผลได้ถูกต้อง

1.2 สามารถเชื่อมต่อและเขียนโปรแกรมควบคุม 7-Segment Display ทั้งแบบธรรมดาและแบบ I2C ได้

1.3 สามารถเชื่อมต่อและเขียนโปรแกรมแสดงผลบน LCD 16x2 แบบ I2C ได้

1.4 สามารถประยุกต์ใช้สวิตช์ควบคุมการแสดงผลได้อย่างเหมาะสม

1.5 มีทักษะการแก้ไขปัญหาและพัฒนาโปรแกรมอย่างเป็นระบบ

**2. อ้างอิงมาตรฐาน/เชื่อมโยงกลุ่มอาชีพ**

**-**

**3. สมรรถนะประจำหน่วย**

3.1 มีทักษะการแก้ไขปัญหาและพัฒนาโปรแกรมอย่างเป็นระบบ

3.2 ทักษะการเขียนโปรแกรมควบคุมอุปกรณ์แสดงผล

3.3 ทักษะการใช้โปรโตคอลการสื่อสาร I2C

3.4 ทักษะการแก้ไขปัญหาและ Debugging

**4. จุดประสงค์เชิงพฤติกรรม**

4.1 อธิบายหลักการทำงานของ 7-Segment Display แบบธรรมดาและแบบ I2C ได้

4.2 อธิบายโครงสร้างและการทำงานของ LCD 16x2 แบบ I2C ได้

4.3 เชื่อมต่อวงจร 7-Segment Display แบบธรรมดากับ ESP32 ได้ถูกต้อง

4.4 เขียนโปรแกรมแสดงผลตัวเลขบน 7-Segment ได้

4.5 เชื่อมต่อและใช้งาน 7-Segment Display แบบ I2C 4 หลักได้

4.6 เชื่อมต่อและเขียนโปรแกรมแสดงผลบน LCD 16x2 แบบ I2C ได้

4.7 ประยุกต์ใช้สวิตช์ควบคุมการแสดงผลได้

**5. เครื่องมือ วัสดุ และอุปกรณ์**

5.1 Arduino IDE 2..3.6

5.2 ชุดฝึกปฏิบัติ

**6. คำแนะนำ/ข้อควรระวัง**

-

**7. ขั้นตอนการปฏิบัติงาน**

7.1 ต่อวงจร LCD 16x2 แบบ I2C เข้ากับ ESP32

7.2 เขียนโปรแกรมให้ LCD 16x2 แบบ I2C ทำงานตามลำดับที่กำหนด

7.3 ทดสอบและปรับแก้โปรแกรม

**8. สรุปและวิจารณ์ผล**

-

**9. การประเมินผล**

1. แบบประเมินสมรรถนะงานภาคปฏิบัติ (6 คะแนน)

ความถูกต้องของการต่อวงจร

ความถูกต้องของโปรแกรม

ความสมบูรณ์ของผลลัพธ์

2. แบบสังเกตพฤติกรรมลักษณะนิสัยการทำงาน (4 คะแนน)

ตรงต่อเวลา

ความร่วมมือและวินัย

**10. เอกสารอ้างอิง /เอกสารค้นคว้าเพิ่มเติม**

-

|  |  |  |
| --- | --- | --- |
|  | **แผนการจัดการเรียนรู้** | หน่วยที่ 3 |
| รหัสวิชา 21901-2007 วิชา เทคโนโลยีระบบสมองกลฝังตัวและไอโอที | สอนครั้งที่ 8 - 12 |
| ชื่อหน่วยการเรียนรู้ การใช้งาน Sensor และ Actuator | ทฤษฎี 4 ชม.  ปฏิบัติ 16 ชม. |
| ชื่อเรื่อง/งาน ใช้งาน Sensor และ Actuator ในการควบคุมระบบอัตโนมัติ | |

**1. ผลลัพธ์การเรียนรู้ระดับหน่วยการเรียน**

1.1 เข้าใจหลักการทำงานของเซนเซอร์ชนิดต่าง ๆ และการประยุกต์ใช้

1.2 ประยุกต์ใช้เซนเซอร์ (LDR, DHT22) ในการวัดและตรวจจับข้อมูลจากสภาพแวดล้อม

1.3 ใช้ Actuator (Relay) ในการควบคุมอุปกรณ์ไฟฟ้า

1.4 ออกแบบและสร้างระบบควบคุมอัตโนมัติโดยใช้เซนเซอร์และ Actuator.

1.5 เขียนโปรแกรมเพื่อประมวลผลข้อมูลจากเซนเซอร์และควบคุม Actuator

1.6 แก้ไขปัญหาและบำรุงรักษาระบบให้ทำงานได้อย่างมีประสิทธิภาพ

**2. อ้างอิงมาตรฐาน/เชื่อมโยงกลุ่มอาชีพ**

**-**

**3. สมรรถนะประจำหน่วย**

3.1 ความรู้ด้านระบบเซนเซอร์: เข้าใจโครงสร้าง หลักการทำงาน และคุณลักษณะของเซนเซอร์ต่าง ๆ

3.2 ความรู้ด้าน Actuator: เข้าใจหลักการทำงานของตัวกระตุ้น โดยเฉพาะ Relay

3.3 ทักษะการออกแบบวงจร: ออกแบบวงจรเชื่อมต่อเซนเซอร์และ Actuator กับไมโครคอนโทรลเลอร์

3.4 ทักษะการเขียนโปรแกรม: เขียนโปรแกรมเพื่อรับค่าจากเซนเซอร์และควบคุม Actuator

3.5 ทักษะการคิดวิจารณญาณ: วิเคราะห์ปัญหาและหาแนวทางแก้ไขอย่างมีเหตุผล

**4. จุดประสงค์เชิงพฤติกรรม**

4.1 บอกความหมายและความสำคัญของเซนเซอร์และ Actuator

4.2 อธิบายหลักการทำงานของ LDR DHT22 และ Relay

4.3 ประกอบวงจรเชื่อมต่อ LDR DHT22 และ Relay

4.4 เขียนโปรแกรมอ่านค่าจาก LDR และ DHT22

4.5 เขียนโปรแกรมควบคุม Relay

4.6 แสดงความสนใจในการเรียนรู้เทคโนโลยีระบบสมองกลฝังตัว

4.7 มีความรับผิดชอบต่อการดูแลอุปกรณ์และวัสดุการสอน

**5. สาระการเรียนรู้**

5.1 เซนเซอร์ (Sensors) ความเป็นมา และความสำคัญของเซนเซอร์ ประเภทของเซนเซอร์ตามประเภทของปริมาณที่วัด การจำแนกเซนเซอร์: Analog และ Digital

5.2 Actuator – Relay ความเป็นมา และความสำคัญของ Actuator ประเภท Actuator: Mechanical Electrical Hydraulic Pneumatic ประเภท Relay: Normally Open (NO) Normally Closed (NC) Changeover

5.3 การสร้างระบบควบคุมอัตโนมัติ ขั้นตอนการออกแบบระบบ การเชื่อมต่อวงจร การเขียนโปรแกรม การทดสอบและแก้ไขปัญหา

**ขั้นสอน (ทฤษฎี 4 ชม.)**

1 บทนำเกี่ยวกับเซนเซอร์ และ Actuator

เนื้อหา:

- ความสำคัญและการประยุกต์ใช้เซนเซอร์และ Actuator ในระบบอัตโนมัติและ IoT

- การจำแนกประเภทเซนเซอร์

- แนวคิดพื้นฐานเกี่ยวกับ Analog และ Digital Signal

- บทบาทของไมโครคอนโทรลเลอร์ในการเชื่อมต่อ

กิจกรรมการสอน:

- บรรยายและอธิบายด้วยสไลด์และรูปภาพ

- แสดงตัวอย่างจริงของเซนเซอร์ประเภทต่าง ๆ

- การอภิปราย: การประยุกต์ใช้เซนเซอร์ในชีวิตประจำวัน

2 เซนเซอร์ LDR และการประยุกต์ใช้งาน

เนื้อหา:

- โครงสร้างและหลักการทำงานของ LDR

- คุณลักษณะทางไฟฟ้า (Resistance Range, Response Time)

- การเชื่อมต่อกับ ADC ของไมโครคอนโทรลเลอร์

- การปรับเทียบและการวัด

กิจกรรมการสอน:

- บรรยายและอธิบายกราฟขอบเขตความจำกัน-ความต้านทานของ LDR

- บรรยายและอธิบายกราฟขอบเขตความจำกัน-ความต้านทานของ LDR

3 เซนเซอร์ DHT22 และการประยุกต์ใช้งาน

เนื้อหา:

- โครงสร้างและหลักการทำงานของ DHT22

- ข้อมูลจำเพาะและค่าสูงสุด-ต่ำสุด

- โปรโตคอลการสื่อสาร DHT22

- การเชื่อมต่อและการใช้ Library

กิจกรรมการสอน:

- บรรยายและอธิบาย Pinout ของ DHT22

- ตัวอย่างกราฟการแสดงผลอุณหภูมิและความชื้น

4 Relay และการออกแบบระบบควบคุม

เนื้อหา:

- โครงสร้างและหลักการทำงานของ Relay

- ประเภท Relay: Single Pole Single Throw (SPST) Single Pole Double Throw (SPDT)

- การควบคุม Relay โดยใช้ GPIO และ Transistor/MOSFET

กิจกรรมการสอน:

- บรรยายและอธิบาย Relay Symbol ในวงจร

- การแสดงการทำงานจริงของ Relay (เสียงคลิก)

**ขั้นปฏิบัติ (16 ชม.)**

ปฏิบัติการที่ 1: การใช้งาน LDR

วัตถุประสงค์: ผู้เรียนสามารถประกอบวงจร LDR อ่านค่าลักษณะของ LDR และเขียนโปรแกรมเพื่อแสดงค่าความสว่าง

ปฏิบัติการที่ 2: การใช้งาน DHT22

วัตถุประสงค์: ผู้เรียนสามารถต่อวงจร DHT22 เขียนโปรแกรมเพื่ออ่านอุณหภูมิและความชื้น

ปฏิบัติการที่ 3: การใช้งาน Relay

วัตถุประสงค์: ผู้เรียนสามารถต่อวงจร Relay ควบคุม Relay โดยใช้โปรแกรมและสามารถควบคุมโคมไฟหรืออุปกรณ์ไฟฟ้า

**7. สื่อและแหล่งการเรียนรู้**

เอกสารประกอบการสอน เรื่อง "การใช้งาน Sensor และ Actuator"

สไลด์นำเสนอ

Video สาธิตการต่อวงจรและการเขียนโปรแกรม

ใบงาน/ใบความรู้

**8. หลักฐานการเรียนรู้**

8.1 หลักฐานความรู้

ใบงานทฤษฎี เรื่อง การใช้งาน Sensor และ Actuator

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

ผู้เรียนส่วนใหญ่เข้าใจหลักการทำงานของคำสั่งพื้นฐานและสามารถประยุกต์ใช้ได้

ผู้เรียนแสดงความสนใจและกระตือรือร้นในการทดลองสร้างรูปแบบการทำงานใหม่ๆ

10.2 ปัญหา อุปสรรคที่พบ

ผู้เรียนบางคนมีพื้นฐานด้านการเขียนโปรแกรมน้อย ทำให้เข้าใจช้ากว่าเพื่อน

เวลาปฏิบัติไม่เพียงพอสำหรับผู้เรียนที่ต้องการทดลองเพิ่มเติม

ผู้เรียนบางคนไม่ส่งรายงานตรงเวลา

10.3 การแก้ไขปัญหา

1) ผลการแก้ไขปัญหาทีส่งผลลัพธ์ที่ดีต่อผู้เรียน

จัดเตรียม Video Tutorial ให้ผู้เรียนสามารถทบทวนได้ด้วยตนเอง

เพิ่มเวลาห้องปฏิบัติการเปิดเพื่อให้ผู้เรียนมาฝึกฝนเพิ่มเติมนอกเวลาเรียน

เน้นย้ำการตรวจสอบวงจรก่อนเปิดไฟทุกครั้ง

2) แนวทางแก้ปัญหาในครั้งต่อไป

พัฒนาสื่อการเรียนรู้ออนไลน์เพิ่มเติม เช่น Interactive Simulation

กำหนดไทม์ไลน์การส่งงานที่ชัดเจนและติดตามอย่างใกล้ชิด

|  |  |  |
| --- | --- | --- |
|  | **ใบความรู้ ที่ 3** | หน่วยที่ 3 |
| รหัสวิชา 21901-2007 วิชา เทคโนโลยีระบบสมองกลฝังตัวและไอโอที | สอนครั้งที่ 8 - 12 |
| ชื่อหน่วยการเรียนรู้ การใช้งาน Sensor และ Actuator | ทฤษฎี 4 ชม.  ปฏิบัติ 16 ชม. |
| ชื่อเรื่อง/งาน ใช้งาน Sensor และ Actuator ในการควบคุมระบบอัตโนมัติ | |

**1. ผลลัพธ์การเรียนรู้ระดับหน่วยการเรียน**

1.1 เข้าใจหลักการทำงานของเซนเซอร์ชนิดต่าง ๆ และการประยุกต์ใช้

1.2 ประยุกต์ใช้เซนเซอร์ (LDR, DHT22) ในการวัดและตรวจจับข้อมูลจากสภาพแวดล้อม

1.3 ใช้ Actuator (Relay) ในการควบคุมอุปกรณ์ไฟฟ้า

1.4 ออกแบบและสร้างระบบควบคุมอัตโนมัติโดยใช้เซนเซอร์และ Actuator.

1.5 เขียนโปรแกรมเพื่อประมวลผลข้อมูลจากเซนเซอร์และควบคุม Actuator

1.6 แก้ไขปัญหาและบำรุงรักษาระบบให้ทำงานได้อย่างมีประสิทธิภาพ

**2. อ้างอิงมาตรฐาน/เชื่อมโยงกลุ่มอาชีพ**

**-**

**3. สมรรถนะประจำหน่วย**

3.1 ความรู้ด้านระบบเซนเซอร์: เข้าใจโครงสร้าง หลักการทำงาน และคุณลักษณะของเซนเซอร์ต่าง ๆ

3.2 ความรู้ด้าน Actuator: เข้าใจหลักการทำงานของตัวกระตุ้น โดยเฉพาะ Relay

3.3 ทักษะการออกแบบวงจร: ออกแบบวงจรเชื่อมต่อเซนเซอร์และ Actuator กับไมโครคอนโทรลเลอร์

3.4 ทักษะการเขียนโปรแกรม: เขียนโปรแกรมเพื่อรับค่าจากเซนเซอร์และควบคุม Actuator

3.5 ทักษะการคิดวิจารณญาณ: วิเคราะห์ปัญหาและหาแนวทางแก้ไขอย่างมีเหตุผล

**4. จุดประสงค์เชิงพฤติกรรม**

4.1 บอกความหมายและความสำคัญของเซนเซอร์และ Actuator

4.2 อธิบายหลักการทำงานของ LDR DHT22 และ Relay

4.3 ประกอบวงจรเชื่อมต่อ LDR DHT22 และ Relay

4.4 เขียนโปรแกรมอ่านค่าจาก LDR และ DHT22

4.5 เขียนโปรแกรมควบคุม Relay

4.6 แสดงความสนใจในการเรียนรู้เทคโนโลยีระบบสมองกลฝังตัว

4.7 มีความรับผิดชอบต่อการดูแลอุปกรณ์และวัสดุการสอน

**5. เนื้อหาสาระ**

**ความสำคัญและการประยุกต์ใช้**

ในยุคปัจจุบัน เทคโนโลยีการออกแบบระบบอัตโนมัติและอินเทอร์เน็ตของสรรพสิ่ง (Internet of Things: IoT) ได้กลายเป็นส่วนสำคัญของชีวิตประจำวันของเรา ทั้งในบ้าน ที่ทำงาน และในโรงงานอุตสาหกรรม เซนเซอร์และตัวกระตุ้น (Actuator) เป็นส่วนประกอบที่สำคัญที่สุดในระบบเหล่านี้

เซนเซอร์ (Sensor) คือ อุปกรณ์ที่ทำหน้าที่รับรู้หรือตรวจจับปริมาณทางกายภาพจากสภาพแวดล้อม เช่น แสง อุณหภูมิ ความชื้น ระยะห่าง และส่งเปลี่ยนปริมาณเหล่านั้นให้เป็นสัญญาณไฟฟ้าที่ไมโครคอนโทรลเลอร์สามารถอ่านและประมวลผลได้

Actuator หรือตัวกระตุ้น คือ อุปกรณ์ที่ทำหน้าที่ขับเคลื่อนหรือควบคุมอุปกรณ์อื่นตามคำสั่งจากไมโครคอนโทรลเลอร์ เช่น เปิดปิดไฟ หมุนมอเตอร์ หรือเปิดปิดวาล์ว

**ความสำคัญของเซนเซอร์และ Actuator มีดังนี้**

1. การตรวจจับข้อมูลแบบอัตโนมัติ: เซนเซอร์ช่วยให้ระบบสามารถตรวจจับสภาพแวดล้อมได้โดยอัตโนมัติ โดยไม่ต้องให้มนุษย์เข้ามาเกี่ยวข้อง

2. การควบคุมที่แม่นยำและเร็ว: Actuator ช่วยให้ระบบสามารถควบคุมอุปกรณ์ต่าง ๆ ได้อย่างแม่นยำและรวดเร็ว

3. ประหยัดพลังงาน: ระบบอัตโนมัติช่วยลดการสิ้นเปลืองพลังงานโดยการควบคุมอุปกรณ์ให้ทำงานเฉพาะเมื่อจำเป็น

4. ความปลอดภัย: ระบบอัตโนมัติช่วยป้องกันอันตรายที่อาจเกิดขึ้นจากการทำงานของมนุษย์

5. ความสะดวกสบาย: ระบบอัตโนมัติทำให้ชีวิตประจำวันของมนุษย์สะดวกขึ้น

**การประยุกต์ใช้ในชีวิตประจำวัน**

**1.** ระบบควบคุมแสงหลักคนอัตโนมัติ: ใช้เซนเซอร์แสง (LDR) ตรวจจับแสงสภาพแวดล้อม เมื่อแสงเข้ามา เซนเซอร์จะส่งสัญญาณให้ไมโครคอนโทรลเลอร์ เพื่อปิดโคมไฟผ่าน Relay

2. ระบบควบคุมเครื่องปรับอากาศอัตโนมัติ: ใช้เซนเซอร์อุณหภูมิและความชื้น (DHT22) ตรวจจับสภาพอากาศ เมื่ออุณหภูมิสูงกว่าที่ตั้ง ระบบจะเปิดเครื่องปรับอากาศโดยอัตโนมัติ

3. ระบบสัญญาณเตือนบ้าน: ใช้เซนเซอร์ตรวจจับการเคลื่อนไหว เมื่อตรวจพบการเคลื่อนไหว ระบบจะส่งสัญญาณเตือนหรือแจ้งเตือนไปยังโทรศัพท์มือถือ

4. ระบบเก็บเกี่ยวข้าวอัตโนมัติในเกษตรกรรม: ใช้เซนเซอร์วัดความชื้นของดินและสภาพอากาศ เพื่อควบคุมระบบรดน้ำและให้ปุ๋ยอัตโนมัติ

**การจำแนกประเภทเซนเซอร์**

เซนเซอร์สามารถจำแนกได้หลายวิธี ตามปริมาณที่วัดและประเภทของสัญญาณที่ออก

การจำแนกตามปริมาณที่วัด

1. Sensor ทางกายภาพ (Physical Sensors):

- เซนเซอร์แสง (LDR, Photodiode)

- เซนเซอร์อุณหภูมิ (Thermistor, DHT22, DS18B20)

- เซนเซอร์ความชื้น (DHT22, Capacitive Humidity Sensor)

- เซนเซอร์ระยะห่าง (Ultrasonic Sensor, IR Sensor)

- เซนเซอร์ความเร่ง (Accelerometer)

2. Sensor ทางเคมี (Chemical Sensors):

- เซนเซอร์ก๊าซ (MQ Series)

- เซนเซอร์คุณภาพน้ำ (pH Sensor, TDS Sensor)

3. Sensor เชิงกล (Mechanical Sensors):

- ปุ่มกด (Push Button)

- เซนเซอร์ความดัน (Pressure Sensor)

**การจำแนกตามประเภทของสัญญาณที่ออก:**

1. Analog Sensor (เซนเซอร์แอนะล็อก):

- ส่งออกสัญญาณแอนะล็อก (ค่าไฟฟ้าต่อเนื่อง) เช่น LDR ส่งออกค่าแรงดัน 0-5V ตามความสว่าง ต้องใช้ ADC (Analog-to-Digital Converter) ในไมโครคอนโทรลเลอร์เพื่อแปลงเป็นค่าดิจิตอล

- Digital Sensor (เซนเซอร์ดิจิตอล) ส่งออกสัญญาณดิจิตอล (ค่า 0 หรือ 1) เช่น DHT22 ส่งออกข้อมูลดิจิตอลแบบอนุกรม สามารถเชื่อมต่อกับ GPIO ของไมโครคอนโทรลเลอร์ได้โดยตรง

**แนวคิดพื้นฐาน: Analog Signal และ Digital Signal**

Analog Signal (สัญญาณแอนะล็อก): สัญญาณแอนะล็อกคือ สัญญาณที่เปลี่ยนแปลงค่าอย่างต่อเนื่องโดยสามารถรับค่าใดๆ ในช่วงที่กำหนด เช่น 0-5V หรือ 0-3.3V

ข้อดีของสัญญาณแอนะล็อก:

- ให้ข้อมูลที่ละเอียดมากขึ้น

- เหมาะสำหรับการวัดปริมาณที่เปลี่ยนแปลงอย่างต่อเนื่อง

ข้อเสียของสัญญาณแอนะล็อก:

- ต้องแปลงเป็นดิจิตอลเพื่อให้ไมโครคอนโทรลเลอร์สามารถประมวลผลได้

- มีความเสี่ยงต่อสัญญาณรบกวน (Noise)

1. Digital Signal (สัญญาณดิจิตอล):

- สัญญาณดิจิตอลคือ สัญญาณที่มีเพียงสองค่า คือ 0 หรือ 1 (Low หรือ High)

ข้อดีของสัญญาณดิจิตอล:

- ง่ายต่อการประมวลผล

- ไม่ได้รับผลกระทบจากสัญญาณรบกวนมากนัก

- สามารถเชื่อมต่อกับ GPIO ได้โดยตรง

ข้อเสียของสัญญาณดิจิตอล:

- ให้ข้อมูลที่น้อยกว่าสัญญาณแอนะล็อก

- ไม่เหมาะสำหรับการวัดปริมาณที่เปลี่ยนแปลงอย่างต่อเนื่อง

**ADC (Analog-to-Digital Converter):**

ADC คือ วงจรที่ทำหน้าที่แปลงสัญญาณแอนะล็อกเป็นสัญญาณดิจิตอล ไมโครคอนโทรลเลอร์ส่วนใหญ่ (เช่น Arduino, Raspberry Pi) มี ADC ในตัว

**เซนเซอร์ LDR (Light Dependent Resistor)**

LDR (Light Dependent Resistor) หรือเซนเซอร์แสงแบบต้านทาน คือ อุปกรณ์ไฟฟ้าที่มีความต้านทานเปลี่ยนแปลงตามปริมาณแสงที่ตกกระทบ เมื่อแสงจ้ามากขึ้น ความต้านทานจะลดลง และเมื่อแสงน้อยลง ความต้านทานจะเพิ่มขึ้น

**LDR มีโครงสร้างอย่างง่าย ประกอบด้วย**

1. แท่งเซมิคอนดักเตอร์ (Semiconductor Bar): ทำจากสารแคดมิอัมซัลไฟด์ (Cadmium Sulphide: CdS) หรือสารอื่นที่มีคุณสมบัติ photoresistive

2. ไฟฟ้าขั้ว (Terminals): ทั้งสองปลายของแท่งเซมิคอนดักเตอร์

3. ชั้นป้องกัน: ส่วนใหญ่มีการเคลือบด้วยสารพอลิเมอร์ เพื่อป้องกันแสงจากด้านข้าง

**ลักษณะภายนอก**

LDR มีลักษณะเป็นอุปกรณ์กลม เส้นผ่านศูนย์กลางประมาณ 5-10 มม. มีสองขา (ไม่มีโพลาริตี้) และหลังผ่องใสเพื่อให้แสงผ่านได้

**หลักการทำงาน**

หลักการทำงานของ LDR นั้นสัมพันธ์กับคุณสมบัติของสารเซมิคอนดักเตอร์ เมื่อไม่มีแสงตกกระทบ (ความมืด) อิเล็กตรอนในสารเซมิคอนดักเตอร์อยู่ในสถานะพื้น อิเล็กตรอนจำนวนน้อยมีพลังงานเพียงพอที่จะเคลื่อนไหว ผลให้ความต้านทานสูงมาก (สูงถึง 1 MΩ หรือมากกว่า) เมื่อมีแสงตกกระทบ โฟตอน (Photon) ของแสงจะให้พลังงานแก่อิเล็กตรอน อิเล็กตรอนได้รับพลังงานพิเพื่อเอาตัวให้พ้นจากสถานะพื้น จำนวนอิเล็กตรอนที่สามารถเคลื่อนไหวเพิ่มขึ้นอย่างมาก ผลให้ความต้านทานลดลง (ลดเหลือ 100 Ω ถึง 1 kΩ) ยิ่งแสงจ้ามากขึ้น อิเล็กตรอนที่มีพลังงานเพียงพอจะมากขึ้น และความต้านทานก็จะลดลงมากขึ้นเท่านั้น

**คุณลักษณะทางไฟฟ้า**

|  |  |
| --- | --- |
| **ลักษณะ** | **ค่า** |
| **ความต้านทานในสภาพมืด (Dark Resistance)** | **1 MΩ - 10 MΩ** |
| **ความต้านทานในสภาพสว่าง (Light Resistance @ 10 lux)** | **5 kΩ - 10 kΩ** |
| **ความต้านทานที่ 100 lux** | **1 kΩ - 5 kΩ** |
| **แรงดันไฟฟ้าสูงสุด** | **100 - 200V** |
| **กำลังไฟฟ้าสูงสุด** | **0.5 - 1 W** |
| **ความเร็วในการตอบสนอง (Response Time)** | **10 ms - 100 ms** |
| **ช่วงความยาวคลื่นที่ไว** | **400 nm - 700 nm (แสงที่มองเห็นได้)** |

**การเชื่อมต่อกับไมโครคอนโทรลเลอร์**

LDR เป็น Analog Sensor ดังนั้นจึงต้องใช้วงจร Voltage Divider (ตัวแบ่งแรงดัน) เพื่อแปลงค่าความต้านทานเป็นค่าแรงดัน

วงจร Voltage Divider แบบพื้นฐาน

**+5V**

**|**

**[R] (ตัวต้านทาน คง ที่ เช่น 10kΩ)**

**|**

**+---o (ADC INPUT)**

**|**

**[LDR] (ความต้านทานเปลี่ยนแปลงตามแสง)**

**|**

**GND**

**การประยุกต์ใช้งาน**

1. ระบบควบคุมแสงอัตโนมัติ (Automatic Lighting Control) วัตถุประสงค์: เปิด/ปิดโคมไฟโดยอัตโนมัติตามแสงสภาพแวดล้อม

2. ระบบรักษาความปลอดภัย (Security System) **วัตถุประสงค์**: ตรวจจับการเคลื่อนไหวในเวลากลางคืน

3. ระบบตรวจจับวัตถุ (Object Detection) **วัตถุประสงค์**: ตรวจจับว่ามีวัตถุผ่านเซนเซอร์หรือไม่

**เซ็นเซอร์ DHT22 (Temperature and Humidity Sensor)**

**ความรู้เบื้องต้นเกี่ยวกับ DHT22**

DHT22 เป็นเซ็นเซอร์วัดอุณหภูมิและความชื้นแบบดิจิทัล ที่ได้รับความนิยมอย่างแพร่หลายในงานด้าน IoT และระบบสมองกลฝังตัว เนื่องจากมีความแม่นยำสูง ราคาไม่แพง และใช้งานง่าย

คุณสมบัติสำคัญของ DHT22

- วัดอุณหภูมิได้ในช่วง -40 ถึง 80 องศาเซลเซียส ความแม่นยำ ±0.5°C

- วัดความชื้นได้ในช่วง 0-100% RH ความแม่นยำ ±2-5% RH

- ใช้แรงดันไฟฟ้า 3.3-5V DC

- ส่งสัญญาณแบบดิจิทัลผ่านขา Data (Single-wire protocol)

- มีความละเอียดสูงกว่า DHT11 โดยสามารถอ่านค่าทศนิยมได้

- เวลาในการอ่านค่า (Sampling Period) อย่างน้อย 2 วินาที

**โครงสร้างและขาสัญญาณของ DHT22**

DHT22 มีขาทั้งหมด 4 ขา (บางรุ่นที่มาพร้อม PCB จะมี 3 ขา) ดังนี้:

- VCC (Pin 1) - ต่อกับไฟเลี้ยง 3.3V หรือ 5V

- DATA (Pin 2) - ขาส่งข้อมูลดิจิทัล ต่อกับขา Digital Pin ของบอร์ด

- NC (Pin 3) - Not Connect ไม่ได้ใช้งาน

- GND (Pin 4) – ต่อกับกราวด์

**หลักการทำงานของ DHT22**

DHT22 ประกอบด้วยเซ็นเซอร์วัดความชื้นแบบ Capacitive และเทอร์มิสเตอร์สำหรับวัดอุณหภูมิ รวมถึงไมโครคอนโทรลเลอร์ภายในที่ทำหน้าที่

- อ่านค่าจากเซ็นเซอร์ทั้งสอง

- แปลงค่าเป็นสัญญาณดิจิทัล

- ส่งข้อมูลออกมาในรูปแบบ Serial Protocol แบบพิเศษ

**การเชื่อมต่อ DHT22 กับบอร์ด ESP32**

**DHT22 ESP32**

**VCC ------> 3.3V**

**DATA ------> Digital Pin (เช่น D4, GPIO4)**

**GND ------> GND**

**\*ต่อตัวต้านทาน 10KΩ ระหว่าง VCC และ DATA**

**การเขียนโปรแกรมควบคุม DHT22**

**#include "DHT.h"**

**// กำหนดขาและประเภทของเซ็นเซอร์**

**#define DHTPIN 4 // ขาที่ต่อกับ DATA ของ DHT22**

**#define DHTTYPE DHT22 // ระบุว่าใช้ DHT22**

**// สร้างออบเจ็กต์ DHT**

**DHT dht(DHTPIN, DHTTYPE);**

**void setup() {**

**Serial.begin(115200);**

**Serial.println("DHT22 Test!");**

**// เริ่มต้นการทำงาน DHT22**

**dht.begin();**

**}**

**void loop() {**

**// รอ 2 วินาที ระหว่างการอ่านค่า**

**delay(2000);**

**// อ่านค่าความชื้น**

**float humidity = dht.readHumidity();**

**// อ่านค่าอุณหภูมิในหน่วยเซลเซียส**

**float temperature = dht.readTemperature();**

**// อ่านค่าอุณหภูมิในหน่วยฟาเรนไฮต์**

**float temperatureF = dht.readTemperature(true);**

**// ตรวจสอบว่าอ่านค่าสำเร็จหรือไม่**

**if (isnan(humidity) || isnan(temperature) || isnan(temperatureF)) {**

**Serial.println("Failed to read from DHT sensor!");**

**return;**

**}**

**// คำนวณ Heat Index**

**float heatIndex = dht.computeHeatIndex(temperatureF, humidity);**

**float heatIndexC = dht.computeHeatIndex(temperature, humidity, false);**

**// แสดงผลข้อมูล**

**Serial.print("Humidity: ");**

**Serial.print(humidity);**

**Serial.print(" %\t");**

**Serial.print("Temperature: ");**

**Serial.print(temperature);**

**Serial.print(" °C / ");**

**Serial.print(temperatureF);**

**Serial.println(" °F");**

**Serial.print("Heat Index: ");**

**Serial.print(heatIndexC);**

**Serial.print(" °C / ");**

**Serial.print(heatIndex);**

**Serial.println(" °F");**

**}**

**Relay Module (อุปกรณ์สวิตช์ควบคุมอัตโนมัติ)**

**ความรู้เบื้องต้นเกี่ยวกับ Relay**

Relay หรือ รีเลย์ เป็นสวิตช์ไฟฟ้าที่ควบคุมด้วยแม่เหล็กไฟฟ้า (Electromagnetic Switch) ใช้สำหรับควบคุมวงจรไฟฟ้ากำลังสูงด้วยสัญญาณควบคุมกำลังต่ำ เป็นอุปกรณ์สำคัญในระบบ Actuator ที่เชื่อมโยงระหว่างระบบสมองกลฝังตัวกับอุปกรณ์ไฟฟ้า เช่น หลอดไฟ พัดลม ปั๊มน้ำ มอเตอร์ เป็นต้น

ประโยชน์หลักของ Relay

- แยกวงจรควบคุม (Low Voltage) ออกจากวงจรกำลัง (High Voltage) เพื่อความปลอดภัย

- ช่วยป้องกันบอร์ด MCU จากกระแสไฟฟ้าสูง

- สามารถควบคุมอุปกรณ์ AC 220V ด้วยสัญญาณ DC 3.3V หรือ 5V

- ควบคุมหลายช่องสัญญาณได้พร้อมกัน (Multi-channel)

**โครงสร้างและหลักการทำงานของ Relay**

**ส่วนประกอบหลักของ Relay**

1. Coil (ขดลวด): เป็นขดลวดแม่เหล็กไฟฟ้า เมื่อมีกระแสไฟฟ้าไหลผ่านจะสร้างสนามแม่เหล็ก

2. Armature (แขนเหล็ก): ชิ้นส่วนที่เคลื่อนที่เมื่อถูกดึงดูดโดยสนามแม่เหล็ก

3. Contact Points (จุดสัมผัส): จุดเชื่อมต่อไฟฟ้า แบ่งเป็น:

- COM (Common): จุดร่วม

- NO (Normally Open): วงจรเปิดตามปกติ จะปิดเมื่อ Relay ทำงาน

- NC (Normally Close): วงจรปิดตามปกติ จะเปิดเมื่อ Relay ทำงาน

**หลักการทำงาน**

1. เมื่อไม่มีกระแสไฟเข้า Coil → Armature อยู่ในตำแหน่งปกติ → COM ต่อกับ NC

2. เมื่อมีกระแสไฟเข้า Coil → เกิดสนามแม่เหล็ก → ดึง Armature → COM ต่อกับ NO

3. เมื่อตัดกระแสไฟ → สนามแม่เหล็กหายไป → Armature กลับสู่ตำแหน่งเดิม

**Relay Module สำหรับระบบสมองกลฝังตัว** Relay Module ที่ใช้กับ Arduino/ESP32 มักมาในรูปแบบบอร์ดสำเร็จรูป ประกอบด้วย

1. Relay 1 ตัวหรือมากกว่า (1, 2, 4, 8 Channel)

2. LED แสดงสถานะการทำงาน

3. วงจรขับ (Driver Circuit) ด้วย Transistor หรือ Optocoupler

4. ป้องกันไฟกระโดน (Flyback Diode)

5. บอร์ด PCB สำเร็จรูปพร้อมขั้วต่อ

**คุณสมบัติทั่วไปของ Relay Module**

1. แรงดันควบคุม: 3.3V - 5V DC

2. แรงดันสวิตช์: AC 250V / DC 30V

3. กระแสสูงสุด: 10A (ขึ้นอยู่กับรุ่น)

4. รองรับทั้ง Active High และ Active Low

**ขาสัญญาณของ Relay Module**

ฝั่งควบคุม (Control Side)

**-** VCC: ต่อกับไฟเลี้ยง 5V (บางรุ่นรองรับ 3.3V)

- GND: ต่อกับกราวด์

- IN1, IN2, IN3...: ขาสัญญาณควบคุม ต่อกับขา Digital Pin ของบอร์ด

ฝั่งสวิตช์ (Switch Side)

- COM: จุดร่วม ต่อกับสายไฟเข้าของอุปกรณ์ที่ต้องการควบคุม

- NO (Normally Open): ต่อกับสายไฟออกของอุปกรณ์ (สำหรับวงจรแบบเปิดตามปกติ)

- NC (Normally Close): ต่อกับสายไฟออกของอุปกรณ์ (สำหรับวงจรแบบปิดตามปกติ)

**Active High vs Active Low**

Active High (ไฮแอคทีฟ)

- Relay ทำงานเมื่อขา IN เป็น HIGH (3.3V/5V)

- Relay ดับเมื่อขา IN เป็น LOW (0V)

- เหมาะสำหรับลอจิกทั่วไป

Active Low (โลว์แอคทีฟ)

- Relay ทำงานเมื่อขา IN เป็น LOW (0V)

- Relay ดับเมื่อขา IN เป็น HIGH (3.3V/5V)

- เป็นแบบที่พบบ่อยใน Relay Module เนื่องจากใช้ Optocoupler

**การเชื่อมต่อ Relay Module**

ฝั่งควบคุม

Relay Module ESP32

VCC ------> 5V

GND ------> GND

IN ------> Digital Pin (เช่น D5, GPIO5)

ฝั่งอุปกรณ์ (ตัวอย่างควบคุมหลอดไฟ AC 220V)

แหล่งจ่ายไฟ AC 220V

|

|-----> COM (Relay)

|

NO (Relay) -----> หลอดไฟ -----> กลับไปแหล่งจ่ายไฟ

**การเขียนโปรแกรมควบคุม Relay**

// กำหนดขา Relay

#define RELAY\_PIN 5

void setup() {

// ตั้งค่าขา Relay เป็น OUTPUT

pinMode(RELAY\_PIN, OUTPUT);

// ตั้งค่าเริ่มต้นให้ Relay ดับ (HIGH สำหรับ Active Low)

digitalWrite(RELAY\_PIN, HIGH);

Serial.begin(115200);

Serial.println("Relay Control Test");

}

void loop() {

// เปิด Relay (Active Low: ส่ง LOW)

Serial.println("Relay ON");

digitalWrite(RELAY\_PIN, LOW);

delay(3000); // เปิด 3 วินาที

// ปิด Relay (Active Low: ส่ง HIGH)

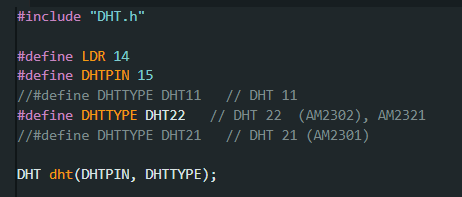
Serial.println("Relay OFF");

digitalWrite(RELAY\_PIN, HIGH);

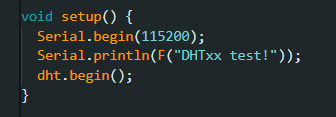
delay(3000); // ปิด 3 วินาที

}

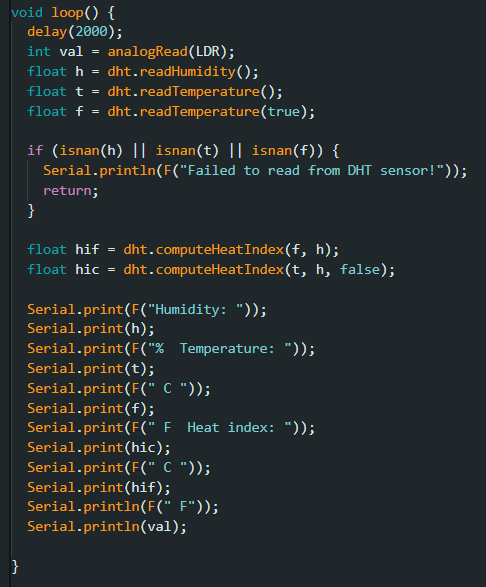
เรียกใช้งานไลบรารีสำหรับสื่อสาร DHT22 และชนิด



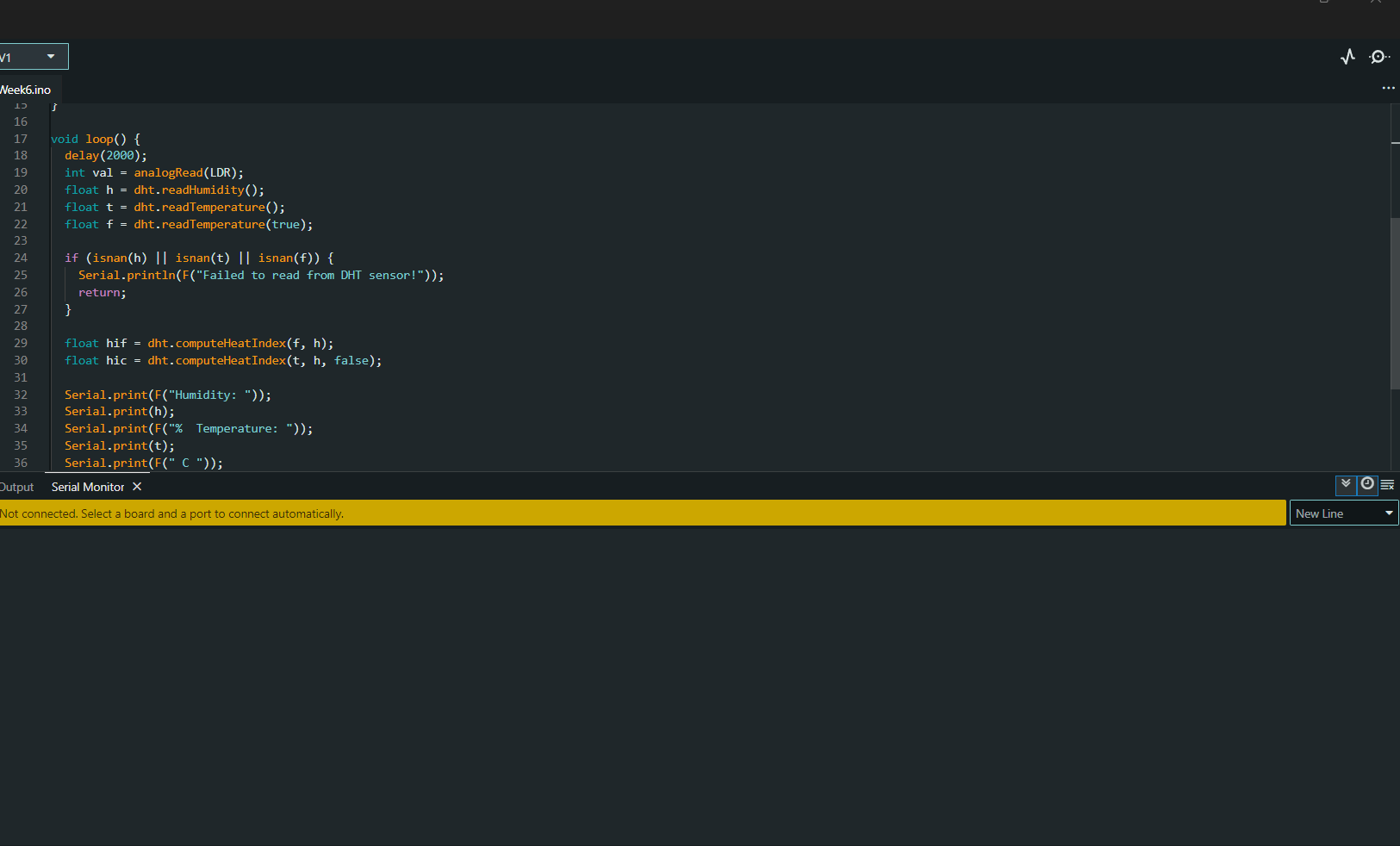
สร้างฟังก์ชันกำหนดรูปแบบการทำงานของพินเชื่อมต่อ



สร้างฟังก์ชันวนซ้ำในการทำงาน

****

ดู Serial monitor ที่เซ็นเซอร์วัดค่าเข้ามาได้

****

**6. แบบฝึกหัด/แบบทดสอบ**

**ใบงาน ที่ 5**

**7. เอกสารอ้างอิง (ขึ้นหน้าใหม่)**

**-**

**8. ภาคผนวก (เฉลยแบบฝึกหัด เฉลยแบบทดสอบ ฯ)**

**-**

|  |  |  |
| --- | --- | --- |
|  | **ใบงาน ที่ 5** | หน่วยที่ 3 |
| รหัสวิชา 21901-2007 วิชา เทคโนโลยีระบบสมองกลฝังตัวและไอโอที | สอนครั้งที่ 8 - 12 |
| ชื่อหน่วยการเรียนรู้ การใช้งาน Sensor และ Actuator | ทฤษฎี 4 ชม.  ปฏิบัติ 16 ชม. |
| ชื่อเรื่อง/งาน ใช้งาน Sensor และ Actuator ในการควบคุมระบบอัตโนมัติ | |

**1. ผลลัพธ์การเรียนรู้ระดับหน่วยการเรียน**

1.1 เข้าใจหลักการทำงานของเซนเซอร์ชนิดต่าง ๆ และการประยุกต์ใช้

1.2 ประยุกต์ใช้เซนเซอร์ (LDR, DHT22) ในการวัดและตรวจจับข้อมูลจากสภาพแวดล้อม

1.3 ใช้ Actuator (Relay) ในการควบคุมอุปกรณ์ไฟฟ้า

1.4 ออกแบบและสร้างระบบควบคุมอัตโนมัติโดยใช้เซนเซอร์และ Actuator.

1.5 เขียนโปรแกรมเพื่อประมวลผลข้อมูลจากเซนเซอร์และควบคุม Actuator

1.6 แก้ไขปัญหาและบำรุงรักษาระบบให้ทำงานได้อย่างมีประสิทธิภาพ

**2. อ้างอิงมาตรฐาน/เชื่อมโยงกลุ่มอาชีพ**

**-**

**3. สมรรถนะประจำหน่วย**

3.1 ความรู้ด้านระบบเซนเซอร์: เข้าใจโครงสร้าง หลักการทำงาน และคุณลักษณะของเซนเซอร์ต่าง ๆ

3.2 ความรู้ด้าน Actuator: เข้าใจหลักการทำงานของตัวกระตุ้น โดยเฉพาะ Relay

3.3 ทักษะการออกแบบวงจร: ออกแบบวงจรเชื่อมต่อเซนเซอร์และ Actuator กับไมโครคอนโทรลเลอร์

3.4 ทักษะการเขียนโปรแกรม: เขียนโปรแกรมเพื่อรับค่าจากเซนเซอร์และควบคุม Actuator

3.5 ทักษะการคิดวิจารณญาณ: วิเคราะห์ปัญหาและหาแนวทางแก้ไขอย่างมีเหตุผล

**4. จุดประสงค์เชิงพฤติกรรม**

4.1 บอกความหมายและความสำคัญของเซนเซอร์และ Actuator

4.2 อธิบายหลักการทำงานของ LDR DHT22 และ Relay

4.3 ประกอบวงจรเชื่อมต่อ LDR DHT22 และ Relay

4.4 เขียนโปรแกรมอ่านค่าจาก LDR และ DHT22

4.5 เขียนโปรแกรมควบคุม Relay

4.6 แสดงความสนใจในการเรียนรู้เทคโนโลยีระบบสมองกลฝังตัว

4.7 มีความรับผิดชอบต่อการดูแลอุปกรณ์และวัสดุการสอน

**5. เครื่องมือ วัสดุ และอุปกรณ์**

5.1 Arduino IDE 2..3.6

5.2 ชุดฝึกปฏิบัติ

**6. คำแนะนำ/ข้อควรระวัง**

-

**7. ขั้นตอนการปฏิบัติงาน**

7.1 ต่อวงจร DHT22 และ LDR หลักเข้ากับ ESP32

7.4 เขียนโปรแกรมรับค่า DHT22 และ LDR ทำงานตามลำดับที่กำหนด

7.5 ทดสอบและปรับแก้โปรแกรม

**8. สรุปและวิจารณ์ผล**

-

**9. การประเมินผล**

1. แบบประเมินสมรรถนะงานภาคปฏิบัติ (6 คะแนน)

ความถูกต้องของการต่อวงจร

ความถูกต้องของโปรแกรม

ความสมบูรณ์ของผลลัพธ์

2. แบบสังเกตพฤติกรรมลักษณะนิสัยการทำงาน (4 คะแนน)

ตรงต่อเวลา

ความร่วมมือและวินัย

**10. เอกสารอ้างอิง /เอกสารค้นคว้าเพิ่มเติม**

-

|  |  |  |
| --- | --- | --- |
|  | **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**

-