|  |  |  |
| --- | --- | --- |
|  | **ชุดการสอน**  **หน่วยสมรรถนะ : UOC 2** อุปกรณ์แสดงผลผลลัพธ์ (Display Device) | **หน่วยที่1** |
| **รหัสวิชา 21901-2007 วิชา เทคโนโลยีระบบสมองกลฝังตัวและไอโอที หน่วยกิต 2** | **สัปดาห์ที่. 3-6** |
| **สมรรถนะย่อย : EOC** การแสดงผลข้อมูลบน 7-Segment และ LCD I²C | **ทฤษฎี 3 ชม.**  **ปฏิบัติ 5 ชม.** |

**1. ผลลัพธ์การเรียนรู้**

ผู้เรียนสามารถเชื่อมต่อและเขียนโปรแกรมควบคุม 7-Segment Display แบบธรรมดา และ LCD/7-Segment I²C ด้วย ESP32 ได้

**2. อ้างอิงมาตรฐานอาชีพ**

-

**3. หน่วยสมรรถนะ : UOC 1** อุปกรณ์แสดงผลผลลัพธ์ (Display Device)

**หน่วยสมรรถนะย่อย : EOC1.1** การแสดงผลข้อมูลบน 7-Segment และ LCD I²C

**4. เกณฑ์การประเมิน (Performance Criteria : PC)**

ผู้เรียนสามารถเชื่อมต่อวงจร ESP32 กับ 7-Segment และ LCD/7-Segment I²C ได้ถูกต้อง (ด้าน Hardware)

**5. ขอบเขตการปฏิบัติ (Range: R)**

การใช้งาน 7-Segment แบบธรรมดา (เชื่อมตรงกับ GPIO) การใช้งาน 7-Segment แบบ I²C (HT16K33) และ การใช้งาน ปุ่มกดดิจิทัล ควบคุมการนับ

**6. จุดประสงค์เชิงพฤติกรรม**

**6.1 เชื่อมต่อวงจร ESP32 + 7-Segment + ปุ่มกด ได้ถูกต้อง**

**6.2 เขียนโปรแกรมแสดงผล ตัวเลข 0–9 บน 7-Segment ได้**

**6.3 ประยุกต์เขียนโปรแกรมควบคุมการนับขึ้น, นับลง, และหยุด/กระพริบ โดยใช้ปุ่มกด**

**7. เนื้อหา**

**1. พื้นฐานการทำงานของ 7-Segment Display**

7-Segment แบบขาธรรมดา (Direct GPIO Control) มีโครงสร้าง: 7-Segment ประกอบด้วย LED 7 ดวง เรียงเป็นรูปตัวเลข 8 (a–g) และอาจมีจุดทศนิยม (dp)

การทำงาน แต่ละ Segment จะติด/ดับ ตามการป้อนไฟฟ้าการแสดงผลตัวเลข 0–9 เกิดจากการเปิด/ปิด Segment ที่กำหนดไว้

เช่น เลข 0 → เปิด a, b, c, d, e, f / ปิด g

การต่อวงจร ต้องต่อสายควบคุมจาก ESP32 ไปยังขา Segment a–g โดยตรง ใช้ digitalWrite() เพื่อกำหนด HIGH/LOW ของแต่ละขา

7-Segment แบบ I²C (HT16K33 Driver) HT16K33 เป็นชิปควบคุม LED/7-Segment ผ่านบัส I²C

ข้อดี

- ใช้เพียง 2 สาย (SDA, SCL) ในการควบคุม ไม่ต้องใช้ GPIO หลายขา

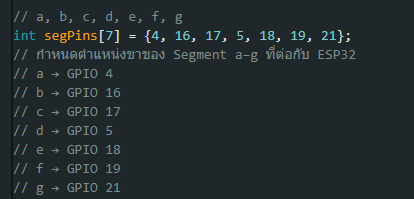
- รองรับ 7-Segment 4 digit, LED Matrix และปุ่มกด

การทำงาน ESP32 ส่งข้อมูลผ่าน I²C Address (0x70) ไปยัง HT16K33 ชิปจะแปลงข้อมูลเป็นการเปิด/ปิด Segment โดยอัตโนมัติ

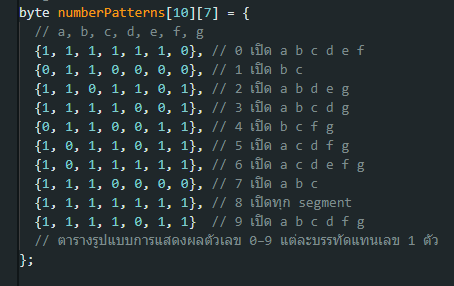
การใช้งานในโค้ด ใช้ Library Adafruit\_GFX และ Adafruit\_LEDBackpack ใช้ฟังก์ชัน matrix.print(num); และ matrix.writeDisplay(); เพื่อแสดงตัวเลข

**2. การอธิบายโค้ดที่ใช้สอน**

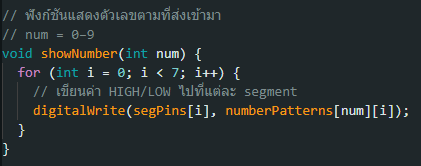
**2.1 พื้นฐานการแสดงผล 0–9 บน 7-Segment ธรรมดา**

****

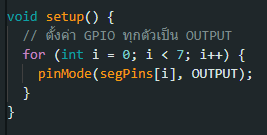
**กำหนดขา ESP32 ที่เชื่อมกับ segment a–g และรูปแบบการเปิด/ปิดแต่ละตัวเลข**

****

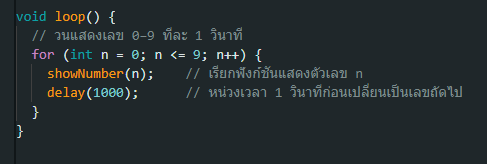
**ฃ ตารางรูปแบบการแสดงผลตัวเลข 0–9 แต่ละบรรทัดแทนเลข 1 ตัว**

****

**ฟังก์ชันแสดงตัวเลขตามที่ส่งเข้ามา**

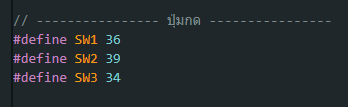
****

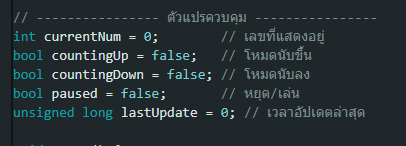
**กำหนดโหมดขาการทำงาน**

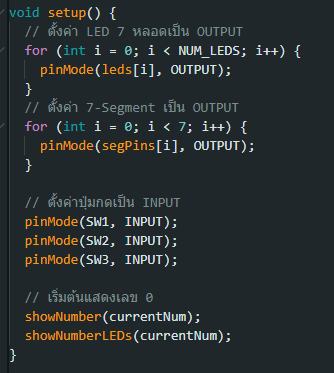
****

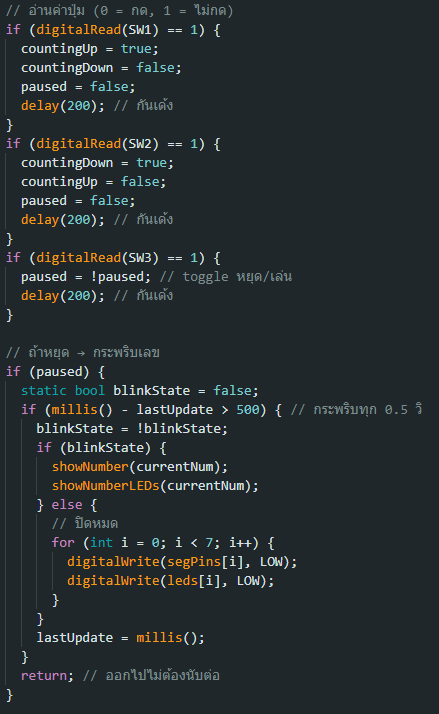
**แสดงผลตัวเลขขึ้นสู่ Segment**

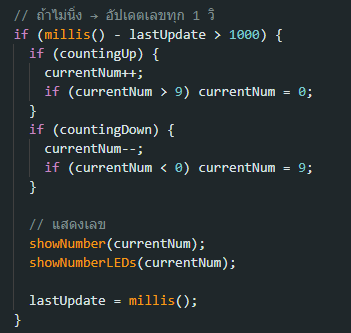
**2.2 เพิ่มปุ่มกดควบคุม Up / Down / Pause**

****

****

****

****

****

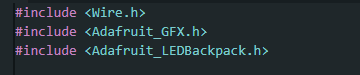
**1. ฟังก์ชัน millis()** millis() เป็นฟังก์ชันใน Arduino ที่ คืนค่าจำนวนมิลลิวินาที (ms) นับตั้งแต่บอร์ดเริ่มทำงาน (reset หรือเปิดเครื่อง)

**2. ตัวแปร lastUpdate** ตัวแปรนี้ไว้เก็บค่าเวลา ครั้งล่าสุดที่อัปเดตการทำงาน เช่น อัปเดตการนับตัวเลขไป 1 ครั้ง ทุกครั้งที่ทำงานเสร็จ เราจะบันทึก lastUpdate = millis(); เพื่อใช้เป็นจุดอ้างอิง

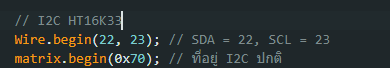
**3. เงื่อนไข if (millis() - lastUpdate > 1000)** millis() = เวลาปัจจุบัน (ms) lastUpdate = เวลาที่อัปเดตครั้งล่าสุด การลบ (millis() - lastUpdate) → จะได้เวลาที่ ผ่านไปแล้ว นับจากการอัปเดตครั้งล่าสุด ถ้าเวลาที่ผ่านไป มากกว่า 1000 ms (1 วินาที) → ให้เข้าไปทำงานใน { ... }

**4. ทำไมไม่ใช้ delay(1000);** delay(1000); จะหยุดโปรแกรมทั้งหมด 1 วินาที → ไม่สามารถตรวจปุ่มกดหรือทำงานอื่น ๆ ได้ แต่ millis() ทำให้สามารถ "นับเวลาไปพร้อมกับตรวจสอบปุ่มกด/ควบคุมอื่น ๆ" ได้ วิธีนี้เรียกว่า Non-blocking Delay หรือ การหน่วงเวลาแบบไม่หยุดโปรแกรม

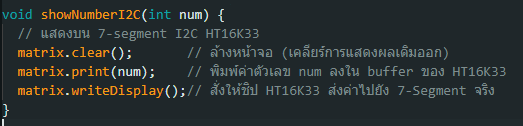
**2.3 เพิ่มการแสดงผลผ่าน 7-Segment I²C HT16K33**

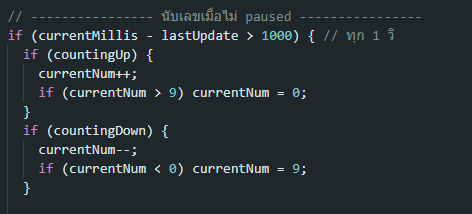


เรียกใช้งานไลบรารีสำหรับ HT16K33



กำหนดขา SDA = 22, SCL = 23 และ I²C Address = 0x70





**8. คู่มือ / Data**

**9. เอกสารอ้างอิง**