maker hub
LEARN • CREATE • CONNECT

Arduino IDE
11 Lab ปฏิบัติการ

เขียนโค้ด C++ ลงบอร์ด · ชุดบอร์ด XIAO ESP32-S3 + AI Expansion

สำหรับ นักเรียนมัธยม (ม.1–ม.6) และครูผู้สอน — ฉบับคู่ขนานกับ “MakerFlow 11 Lab” ใช้โจทย์ อุปกรณ์ ขา และเกณฑ์ชุดเดียวกัน เหมาะสำหรับต่อยอดหลังเรียนแบบลากวางแล้ว หรือสอนกลุ่มที่พร้อมเขียนโค้ดตั้งแต่แรก

ใช้โปรแกรม Arduino IDE 2 (ฟรี) บน Windows / macOS / Linux · แพ็กเกจบอร์ด esp32 by Espressif · ไลบรารี Adafruit SH110X (สำหรับจอ OLED)

อุปกรณ์ XIAO ESP32-S3 Sense · บอร์ดขยาย AI Expansion (รีเลย์ 2 ช่อง) · จอ OLED · SHT30 · BH1750 · MPU6050 · เซนเซอร์ความชื้นในดิน

โค้ดทุกไฟล์ในเล่มนี้คอมไพล์ผ่านจริงด้วย arduino-cli (บอร์ด XIAO_ESP32S3 · esp32 3.3.10) โดยไม่มีคำเตือน · ขา เซนเซอร์ และสูตรแปลงค่าตรงกับเฟิร์มแวร์ MakerFlow ที่ทดสอบกับคิทจริงแล้ว · **ตัวสเก็ตช์ Arduino เองยังไม่ได้อัปโหลดลงบอร์ดจริง** — ครูควรลองก่อนสอน (ดูภาคผนวก ค)

จัดทำโดย บริษัท เมคเกอร์ฮับ จำกัด (MakerHub Co., Ltd.) · 30 กันยายน 2569

สารบัญ

Lab	หัวข้อ	เวลา	ไฟล์
Lab 1	ไฟกะพริบ — โปรแกรมแรกของบอร์ด	20 นาที	`Lab01_Blink`
Lab 2	ตัวนับ — ดูค่าบนคอมพิวเตอร์และบนจอ OLED	20 นาที	`Lab02_Counter`
Lab 3	อ่านอุณหภูมิและความชื้นด้วย SHT30	20 นาที	`Lab03_SHT30`
Lab 4	แสดงอุณหภูมิบนจอ OLED	15 นาที	`Lab04_OLED`
Lab 5	คำนวณ — แปลงองศาเซลเซียสเป็นฟาเรนไฮต์	20 นาที	`Lab05_Fahrenheit`
Lab 6	ตัดสินใจ — ไฟเตือนเมื่อร้อนเกินกำหนด	20 นาที	`Lab06_HeatAlarm`
Lab 7	รีเลย์ 2 ช่อง — พัดลมและเครื่องลดความชื้นอัตโนมัติ	25 นาที	`Lab07_Relays`
Lab 8	ไฟส่องสว่างอัตโนมัติด้วย BH1750	20 นาที	`Lab08_AutoLight`
Lab 9	รดน้ำต้นไม้อัตโนมัติด้วยเซนเซอร์ความชื้นในดิน	30 นาที	`Lab09a_SoilRaw` · `Lab09b_SoilPump`
Lab 10	สัญญาณเตือนการเอียงด้วย MPU6050	25 นาที	`Lab10_Tilt`
Lab 11	โปรเจกต์รวม — สถานีตรวจอากาศห้องเรียน	40 นาที	`Lab11_Station`

เวลาในตารางอิงฉบับ MakerFlow — ฉบับโค้ดควรเผื่อเวลาเพิ่ม 30–50 % สำหรับการพิมพ์โค้ดและอัปโหลด (ครั้งละ ~20–40 วินาที) · ครูแจกไฟล์ .ino ให้เปิดแล้วแก้ต่อได้ ประหยัดเวลาพิมพ์

อุปกรณ์ในชุด

อุปกรณ์	หน้าที่	ที่อยู่ I2C
XIAO ESP32-S3 Sense	สมองของชุด · ไฟ LED บนบอร์ด (ขา LED) · กล้อง	—
บอร์ดขยาย AI Expansion	รีเลย์ 2 ช่อง (ช่อง 1 = D0, ช่อง 2 = D1) · ขั้ว Grove I2C (D4/D5) · RS485	—
จอ OLED 1.3 นิ้ว (SH1106)	แสดงตัวเลข/ข้อความอังกฤษ	0x3C
SHT30	อุณหภูมิ + ความชื้นอากาศ	0x44
BH1750	ความสว่าง (lux)	0x23
MPU6050	ความเร่ง 3 แกน / ไจโร	0x68
เซนเซอร์ความชื้นในดิน (แอนะล็อก)	ความชื้นในดิน — Lab 9 (ใบงานเสริม)	ขา D3

เซนเซอร์และจอทั้งหมดใช้ **สาย I2C คู่เดียวกัน** (ขั้ว Grove บนบอร์ดขยาย = ขา D4/SDA · D5/SCL) ต่อพ่วงกันผ่าน Grove Hub ได้ เพราะที่อยู่ไม่ซ้ำกัน · ในโค้ดเรียก `Wire.begin()` เฉย ๆ บอร์ดรู้เองว่าใช้ขา D4/D5

> ⚠️ **อ่านก่อนเริ่ม** — การอัปโหลดโค้ด Arduino **จะเขียนทับเฟิร์มแวร์ MakerFlow** บนบอร์ด ถ้าจะกลับไปใช้ MakerFlow ต้องติดตั้งเฟิร์มแวร์ใหม่จากหน้า maker-hub.net/makerflow (ราว 1 นาที ดูภาคผนวก ก) — วางแผนคาบให้ไม่สลับไปมาบ่อย

ตั้งค่าครั้งแรก (ทำครั้งเดียวต่อเครื่อง)

ต้องต่ออินเทอร์เน็ตตอนติดตั้ง (แพ็กเกจบอร์ด ESP32 ราว 300–500 MB) — แนะนำให้ครูติดตั้งทุกเครื่องไว้ล่วงหน้า **ก่อนวันสอน**

1. ติดตั้ง Arduino IDE 2

1. ดาวน์โหลดจาก **arduino.cc/en/software** เลือกระบบปฏิบัติการของเครื่อง แล้วติดตั้งตามปกติ
2. Windows: ถ้ามีหน้าต่างถามติดตั้ง *driver* ให้กด Install ทุกครั้ง

2. เพิ่มบอร์ด ESP32

1. เมนู **File → Preferences** ช่อง **Additional boards manager URLs** วาง `https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json` แล้วกด OK
2. แถบซ้ายกดไอคอน **Boards Manager** (รูปบอร์ด) ค้นหา **esp32** → ตัวที่เขียนว่า **esp32 by Espressif Systems** → Install (รุ่น 3.x)

3. ติดตั้งไลบรารีจอ OLED

1. แถบซ้ายกดไอคอน **Library Manager** (รูปหนังสือ) ค้นหา **Adafruit SH1106X** → Install
2. มีหน้าต่างถามติดตั้งไลบรารีที่ต้องใช้คู่กัน (Adafruit GFX · Adafruit BusIO) ให้กด **Install All**
3. SHT30 · BH1750 · MPU6050 **ไม่ต้องติดตั้งไลบรารี** — ใบงานเขียนโค้ดคุยกับเซนเซอร์เองผ่าน `Wire` ซึ่งมีมาในตัว (ได้เรียนรู้ว่าเซนเซอร์ทำงานอย่างไรด้วย)

4. เลือกบอร์ดและพอร์ต (ทำทุกครั้งที่เปิดโปรแกรม)

1. เสียบสาย USB-C เข้าบอร์ด
2. เมนู **Tools → Board → esp32 → XIAO_ESP32S3**
3. เมนู **Tools → Port** เลือกพอร์ตที่โผล่มา (Windows = COM*x* · macOS = /dev/cu.usbmodem… · Linux = /dev/ttyACM0)
4. ตรวจ **Tools → USB CDC On Boot = Enabled** (ค่าเริ่มต้นอยู่แล้ว — ถ้าเป็น Disabled Serial Monitor จะว่าง)

5. ทดลอง: อัปโหลดโปรแกรมแรก

เปิดไฟล์ `Lab01_Blink.ino` กด **→ Upload** — **บอร์ดที่ยังมีเฟิร์มแวร์ MakerFlow อยู่ หรืออัปโหลดครั้งแรก อาจต้องเข้าโหมดดาวน์โหลดก่อน:** ถอดสาย → กดปุ่ม **B** (BOOT) บนตัว XIAO ค้างไว้ → เสียบสาย → ปล่อยปุ่ม → เลือกพอร์ตใหม่ → Upload → เสร็จแล้วกดปุ่ม **R** (RESET)

รู้จักหน้าจอ Arduino IDE

ส่วน	ใช้ทำอะไร
✔ Verify	ตรวจและแปลงโค้ด (คอมไพล์) โดยยังไม่ส่งลงบอร์ด — ใช้หาจุดผิดเร็ว ๆ
→ Upload	คอมไพล์แล้วส่งลงบอร์ด
ช่องเลือกบอร์ด (แถบบน)	ต้องขึ้น XIAO_ESP32S3 และพอร์ตที่ถูก
พื้นที่เขียนโค้ด (กลาง)	พิมพ์โค้ด · **Ctrl+T** จัดย่อหน้าให้อัตโนมัติ · **Ctrl+/** ทำบรรทัดเป็นคอมเมนต์
Output (ล่าง)	ผลคอมไพล์/อัปโหลด — **ข้อความสีแดง** บอกบรรทัดที่ผิด อ่านบรรทัดแรกที่ขึ้น `error:` ก่อน
🔍 Serial Monitor (ขวาบน)	ดูข้อความที่บอร์ดพิมพ์ด้วย `Serial.print` · ตั้ง 115200 baud
Tools → Serial Plotter	วาดตัวเลขที่บอร์ดพิมพ์เป็นกราฟ

จากบล็อกสู่โค้ด

ถ้านักเรียนเคยเรียนฉบับ MakerFlow มาแล้ว ทุกบล็อกมีคำสั่ง Arduino ที่ทำหน้าที่เดียวกัน (ปุ่ม **</> โค้ด** ใน MakerFlow ก็แปลงให้ดูได้ — แต่โค้ดในเล่มนี้เขียนด้วยมือให้อ่านง่ายสำหรับผู้เริ่มต้น)

บล็อก MakerFlow	โค้ด Arduino	ใช้ใน Lab
⏱ ทุก ๆ N มิลลิวินาที	`delay(N)` หรือ `millis()`	1, 11
💡 สั่งขาดิจิทัล	`pinMode(ขา, OUTPUT)` + `digitalWrite(ขา, HIGH/LOW)`	1, 6, 10
🔢 ตัวนับ	ตัวแปร `int` + `count = count + 1`	2
📟 แสดงค่าบนหน้าจอ	`Serial.print / println` → Serial Monitor	ทุก Lab
📺 จอ OLED	ไลบรารี Adafruit SH110X	2, 4, 5, 8, 9, 11
🌡 SHT30 · 🔆 BH1750 · 🎯 MPU6050	ไลบรารี `Wire` (I2C) + ฟังก์ชันที่เราเขียน	3–8, 10, 11
🧮 คำนวณ	`+ - * /` · `fabs()` · `round()`	5, 10
⚖️ เปรียบเทียบ	`> < >= <= == !=` → ได้ `bool`	6–11
🔗 ตรรกะ และ/หรือ	`&&` · `||` · `!`	11
🔌 รีเลย์บนแผงขยาย	`digitalWrite(D0 หรือ D1, HIGH)`	7, 8, 9, 11
📈 อ่านค่าแอนะล็อก	`analogRead(ขา)` → 0–4095	9
📋 ลำดับขั้นตอน	เขียนคำสั่งเรียงกัน + `for`	11

โครงของทุกโปรแกรม

```
#include <Wire.h>              // 1. นำไลบรารีมาใช้

const int WAIT_MS = 500;        // 2. ค่าคงที่ / ตัวแปรที่ใช้ทั้งโปรแกรม

void setup() {                  // 3. ทำครั้งเดียวตอนเปิดเครื่อง
  Serial.begin(115200);
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {                   // 4. ทำซ้ำไปเรื่อย ๆ ไม่มีวันจบ
  // อ่านเซนเซอร์ → คิด → สั่งงาน
}
```

กติกาภาษา C++ ที่มือใหม่พลาดบ่อย

- จบทุกคำสั่งด้วย `;` · วงเล็บปีกกา `{ }` ต้องมีคู่เสมอ
- ตัวพิมพ์เล็ก-ใหญ่ **ต่างกัน**: `digitalWrite` ถูก · `digitalwrite` ผิด
- `=` คือ “ใส่ค่า” · `==` คือ “ถามว่าเท่ากันไหม”
- ข้อความภาษาไทยใส่ได้ใน `"…"` และคอมเมนต์ `// …` เท่านั้น — ชื่อตัวแปรต้องเป็นอังกฤษ
- ตัวเลขทศนิยมต้องมีจุด: `175.0 / 65535` ≠ `175 / 65535` (แบบหลังได้ 0)

> ⚠️ **กติกาความปลอดภัยของห้องปฏิบัติการ** — ใช้ไฟจากสาย USB หรือแหล่งจ่ายไฟต่ำเท่านั้น · **ห้ามต่อไฟบ้าน 220 V** กับบอร์ดหรือรีเลย์ · ถอดสาย USB ก่อนเสียบ/ถอดสายเซนเซอร์ · ถ้าได้กลิ่นไหม้หรือบอร์ดร้อนผิดปกติ ถอดสายแล้วเรียกครูทันที

LAB 1 ไฟกะพริบ — โปรแกรมแรกของบอร์ด

✅ คอมไพล์ผ่าน | ยังไม่ได้อัปโหลดลงบอร์ดจริง | เวลาโดยประมาณ 20 นาที · ไฟล์ `Lab01_Blink.ino`

🌏 **ใช้ที่ไหนในชีวิตจริง?** ไฟเลี้ยวรถยนต์ · ไฟกะพริบบนเราเตอร์ Wi-Fi · ไฟสัญญาณคนข้ามถนน — ทั้งหมดคือ “ไฟกะพริบตามเวลา” แบบเดียวกับที่เราจะทำ

🎯 เป้าหมาย

- รู้จักหน้าจอ Arduino IDE และโครง `setup()` / `loop()`
- ใช้คำสั่ง `pinMode` · `digitalWrite` · `delay`
- อัปโหลดโปรแกรมแรกลงบอร์ด

🧰 อุปกรณ์

บอร์ด XIAO ESP32-S3 (ใช้ไฟ LED สีส้มบนตัวบอร์ด) · สาย USB-C

💭 **แนวคิด** — ทุกโปรแกรม Arduino มีสองห้อง: **`setup()`** ทำ *ครั้งเดียว* ตอนเปิดเครื่อง (เหมือนจัดโต๊ะก่อนเริ่มงาน) และ **`loop()`** ทำ *ซ้ำไม่รู้จบ* (เหมือนเข็มนาฬิกาที่เดินวนไปเรื่อย ๆ) · Lab นี้ใน loop มีแค่ 4 บรรทัด: เปิดไฟ → รอ → ปิดไฟ → รอ แล้ววนกลับขึ้นบนเอง

🧩 เทียบกับบล็อก MakerFlow

บล็อกในฉบับ MakerFlow	เขียนเป็นโค้ด Arduino
⏱ ทุก ๆ N มิลลิวินาที	`delay(WAIT_MS)` ใน `loop()`
🔀 สลับเปิด / ปิด	เขียนสองจังหวะตรง ๆ: LOW แล้ว HIGH
💡 สั่งขาดิจิทัล + กลับค่า	`digitalWrite(LED_BUILTIN, LOW)` = ไฟติด

⌨️ พิมพ์โค้ดนี้ (หรือเปิดไฟล์ `Lab01_Blink.ino`)

```
// Lab 1 — ไฟกะพริบ: โปรแกรมแรกของบอร์ด XIAO ESP32-S3
// ไฟ LED สีส้มบนบอร์ดต่อแบบ active-low: สั่ง LOW = ไฟติด · สั่ง HIGH = ไฟดับ

const int WAIT_MS = 500;          // ติดกี่มิลลิวินาที ดับกี่มิลลิวินาที

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);   // บอกบอร์ดว่าขา LED ใช้ "ส่งไฟออก"
}

void loop() {
  digitalWrite(LED_BUILTIN, LOW);   // ไฟติด (active-low)
  delay(WAIT_MS);
  digitalWrite(LED_BUILTIN, HIGH);  // ไฟดับ
  delay(WAIT_MS);
}
```

🔍 บรรทัดสำคัญ

โค้ด	ความหมาย
`pinMode(LED_BUILTIN, OUTPUT);`	บอกบอร์ดว่าขา LED ใช้ **ส่งไฟออก** (ถ้าไม่บอก ไฟจะไม่ติดหรือติดหรี่ ๆ)
`digitalWrite(LED_BUILTIN, LOW);`	ไฟ LED บน XIAO ต่อแบบ *active-low* — สั่ง LOW แล้วไฟ **ติด**
`delay(WAIT_MS);`	หยุดรอ 500 มิลลิวินาที ระหว่างนี้บอร์ดไม่ทำอะไรเลย

> 🔮 **ลองทำนายก่อนอัปโหลด** — `WAIT_MS = 500` — ใน 10 วินาที ไฟจะ **ติด** กี่ครั้ง?
> คำทำนายของฉัน:
> ---

▶️ อัปโหลดลงบอร์ด

กดปุ่ม → **Upload** (ลูกศรขวา มุมซ้ายบน) รอจนขึ้น **Done uploading** → เปิด **Serial Monitor** (ไอคอนแว่นขยายมุมขวาบน) ตั้ง **115200 baud**

👀 ผลที่ควรเห็น

- ไฟสีส้มบนบอร์ดติดครึ่งวินาที ดับครึ่งวินาที สลับไปเรื่อย ๆ
- มุมขวาล่างของ Arduino IDE ขึ้น **Done uploading** ก่อนไฟเริ่มกะพริบ

📝 บันทึกผลของฉัน

WAIT_MS (มิลลิวินาที)	ไฟติดกี่ครั้งใน 10 วินาที	ตามองทันไหม
500		
200		
100		
20		

> 💡 **รู้ไหม? ทำไม LOW = ติด?** ขา LED ของ XIAO ต่ออยู่กับไฟ 3.3 V อีกด้านหนึ่ง ไฟจะไหลผ่านหลอดได้ก็ต่อเมื่อขาเป็น 0 V (LOW) — กลับด้านกับที่เราคิด เหมือนสวิตช์ที่ติดตั้งกลับหัว · ในฉบับ MakerFlow คือการติ๊ก “กลับค่า” นั่นเอง

🚀 ท้าทายตัวเอง

⭐ เปลี่ยน `WAIT_MS` เป็น 100 — ไฟเร็วขึ้นแค่ไหน?

⭐⭐ ทำให้ไฟ **ติดสั้น ดับนาน** (ติด 100 ดับ 900) เหมือนไฟกระพริบบนเราเตอร์ — ต้องแยกตัวแปรอย่างไร?

⭐⭐⭐ ทำสัญญาณ **SOS** (สั้น 3 ยาว 3 สั้น 3) ด้วยคำสั่ง `for` ให้โค้ดสั้นที่สุด

LAB 2 ตัวนับ — ดูค่าบนคอมพิวเตอร์และบนจอ OLED

✅ คอมไพล์ผ่าน | ยังไม่ได้อัปโหลดลงบอร์ดจริง | เวลาโดยประมาณ 20 นาที · ไฟล์ `Lab02_Counter.ino`

🌏 **ใช้ที่ไหนในชีวิตจริง?** เครื่องนับคนเข้าห้าง · สกอร์บอร์ดกีฬา · มิเตอร์แท็กซี่ · นาฬิกาจับเวลา

🎯 เป้าหมาย

- ใช้ **ตัวแปร** เก็บค่าที่เปลี่ยนไปเรื่อย ๆ และใช้ `if` ครั้งแรก
- พิมพ์ค่าออก **Serial Monitor** บนคอมพิวเตอร์
- ติดตั้งไลบรารีและใช้จอ OLED ครั้งแรก

🧰 อุปกรณ์

บอร์ด + จอ OLED 1.3 นิ้ว (SH1106) ต่อช่อง I2C (Grove)

💭 **แนวคิด — ตัวแปร** `count` คือกล่องใส่ตัวเลขที่อยู่ในบอร์ด · ทุกรอบของ loop เราหยิบค่าเดิมมา +1 แล้วใส่กลับ และถ้าถึง 10 ก็เทกล่องทิ้งให้เป็น 0 · ค่าเดียวกันส่งไปได้หลายที่ — บรรทัดหนึ่งพิมพ์ขึ้นคอม อีกชุดวาดขึ้นจอ OLED

🧩 เทียบกับบล็อก MakerFlow

บล็อกในฉบับ MakerFlow	เขียนเป็นโค้ด Arduino
⏱ ทุก ๆ 1000	`delay(1000)` ท้าย loop
🔢 ตัวนับ (วนกลับเมื่อถึง 10)	`count = count + 1; if (count >= 10) count = 0;`
🖥 แสดงค่าบนหน้าจอ	`Serial.print(…)` → ดูใน Serial Monitor
📺 จอ OLED	ไลบรารี Adafruit SH110X: `clearDisplay · print · display`

⌨️ พิมพ์โค้ดนี้ (หรือเปิดไฟล์ `Lab02_Counter.ino`)

```
// Lab 2 — ตัวนับ: ดูค่าบนคอมพิวเตอร์ (Serial Monitor) และบนจอ OLED
// ต้องติดตั้งไลบรารี "Adafruit SH110X" ก่อน (Sketch → Include Library → Manage Libraries…)
#include <Wire.h>
#include <Adafruit_SH110X.h>

Adafruit_SH1106G oled(128, 64, &Wire, -1);   // จอ 1.3 นิ้วในชุดใช้ชิป SH1106

int count = 0;

void setup() {
  Serial.begin(115200);
  Wire.begin();                    // สาย I2C: SDA = D4 · SCL = D5 (ขั้ว Grove)
  oled.begin(0x3C, true);          // ที่อยู่ของจอบนสาย I2C
}

void loop() {
  count = count + 1;               // เพิ่มครั้งละ 1
  if (count >= 10) count = 0;      // ถึง 10 เมื่อไร วนกลับเป็น 0

  Serial.print("นับได้ = ");
  Serial.println(count);

  oled.clearDisplay();
  oled.setTextColor(SH110X_WHITE);
  oled.setTextSize(1);
  oled.setCursor(0, 0);
  oled.print("Count");             // จอรู้จักแค่ตัวอักษรอังกฤษ
  oled.setTextSize(3);
  oled.setCursor(0, 20);
  oled.print(count);
  oled.display();                  // สั่งให้ภาพที่วาดไว้ขึ้นจอจริง
```

```
  delay(1000);
}
```

🔍 บรรทัดสำคัญ

โค้ด	ความหมาย
`int count = 0;`	สร้างตัวแปรไว้ **นอก** loop — ถ้าสร้างใน loop มันจะถูกรีเซ็ตเป็น 0 ทุกรอบ
`Serial.begin(115200);`	เปิดช่องคุยกับคอม ความเร็ว 115200 — Serial Monitor ต้องตั้งเลขเดียวกัน
`oled.begin(0x3C, true);`	เริ่มจอที่ที่อยู่ 0x3C บนสาย I2C
`oled.display();`	วาดทุกอย่างไว้ในหน่วยความจำก่อน บรรทัดนี้จึงค่อยส่งขึ้นจอจริง — **ลืมบรรทัดนี้ = จอว่าง**

> 🔮 **ลองทำนายก่อนอัปโหลด** — ใช้เงื่อนไข `if (count >= 10) count = 0;` — ตัวเลขที่ **มากที่สุด** ที่จะเห็นบนจอคือ 10 หรือ 9?
>
> คำทำนายของฉัน:
> ---

▶️ อัปโหลดลงบอร์ด

กดปุ่ม → **Upload** (ลูกศรขวา มุมซ้ายบน) รอจนขึ้น **Done uploading** → เปิด **Serial Monitor** (ไอคอนแว่นขยายมุมขวาบน) ตั้ง **115200 baud**

👀 ผลที่ควรเห็น

- จอ OLED แสดง **Count** ตามด้วยตัวเลขที่เพิ่มขึ้นทุกวินาที ถึง 9 แล้วกลับเป็น 0
- Serial Monitor (ตั้ง 115200 baud) ขึ้น `นับได้ = 1`, `นับได้ = 2`, …

📝 บันทึกผลของฉัน

วินาทีที่	1	2	3	4	5	6	7	8	9	10	11	12
ตัวเลขบนจอ												

> 💡 **รู้ไหม? ไลบรารี (library)** คือโค้ดสำเร็จรูปที่คนอื่นเขียนไว้ให้ยืมใช้ — การวาดตัวอักษรลงจอต้องใช้โค้ดหลายร้อยบรรทัด เราแค่ `#include` แล้วเรียก `print` · จอในชุด **รู้จักแค่ภาษาอังกฤษ** ส่วน Serial Monitor บนคอมใช้ภาษาไทยได้ปกติ

🚀 ท้าทายตัวเอง

⭐ ให้นับทีละ 5 และวนกลับเมื่อถึง 100

⭐⭐ ให้นับ **ถอยหลัง** 10 → 0 แล้วขึ้นคำว่า `GO!` บนจอ

⭐⭐⭐ ทำนาฬิกาจับเวลา `mm:ss` บนจอ (ใบ้: นาที = count / 60 · วินาที = count % 60 และเติม 0 ข้างหน้าเมื่อน้อยกว่า 10)

LAB 3 อ่านอุณหภูมิและความชื้นด้วย SHT30

✅ คอมไพล์ผ่าน | ยังไม่ได้อัปโหลดลงบอร์ดจริง | เวลาโดยประมาณ 20 นาที · ไฟล์ `Lab03_SHT30.ino`

🌏 **ใช้ที่ไหนในชีวิตจริง?** เครื่องปรับอากาศ · แอปพยากรณ์อากาศ · โรงเรือนปลูกผัก · ตู้ฟักไข่ — ทุกตัวต้องรู้อุณหภูมิและความชื้นก่อนตัดสินใจ

🎯 เป้าหมาย

- เข้าใจว่าบอร์ด **คุยกับเซนเซอร์** ผ่านสาย I2C อย่างไร (ส่งคำสั่ง → รอ → อ่านผล)
- เขียน/เรียกใช้ **ฟังก์ชัน** ของเราเอง
- อ่านอุณหภูมิและความชื้นจริง

🧰 อุปกรณ์

บอร์ด + เซนเซอร์ SHT30 (ที่อยู่ 0x44) ต่อช่อง I2C

💭 **แนวคิด** — การอ่าน SHT30 เหมือน **สั่งอาหาร**: บอกร้าน (ที่อยู่ 0x44) ว่าเอาเมนูอะไร (คำสั่ง 0x24 0x00) → รอทำ 20 มิลลิวินาที → รับอาหาร 6 ไบต์ · เซนเซอร์ส่งมาเป็นเลขดิบ 0–65535 เราต้องแปลงเป็นองศาด้วยสูตรของผู้ผลิต · ทั้งหมดรวมไว้ในฟังก์ชัน `readSHT30()` เรียกใช้ได้ด้วยบรรทัดเดียว (Lab ถัดไปก็ใช้ฟังก์ชันนี้ซ้ำ)

🧩 เทียบกับบล็อก MakerFlow

บล็อกในฉบับ MakerFlow	เขียนเป็นโค้ด Arduino
🌡️ SHT30 (ที่อยู่ 0x44)	ฟังก์ชัน `readSHT30(t, h)` ที่คุยผ่าน `Wire`
อ่านทุก 1000	`delay(1000)`
📟 แสดงค่า ×2	`Serial.println(t, 2)` · `Serial.println(h, 2)`

⌨️ พิมพ์โค้ดนี้ (หรือเปิดไฟล์ `Lab03_SHT30.ino`)

```
// Lab 3 — อ่านอุณหภูมิและความชื้นด้วย SHT30 (ที่อยู่ 0x44)
// คุยกับเซนเซอร์ตรง ๆ ผ่านสาย I2C — ไม่ต้องติดตั้งไลบรารีเพิ่ม
#include <Wire.h>

const uint8_t SHT30_ADDR = 0x44;

// สั่งวัด 1 ครั้ง แล้วอ่านผล 6 ไบต์ · คืน true ถ้าอ่านสำเร็จ
bool readSHT30(float &tempC, float &humid) {
  Wire.beginTransmission(SHT30_ADDR);
  Wire.write(0x24);                // คำสั่ง "วัดครั้งเดียว ความละเอียดสูง"
  Wire.write(0x00);
  if (Wire.endTransmission() != 0) return false;   // ไม่มีใครตอบ = สายหลุด/ที่อยู่ผิด
  delay(20);                       // เซนเซอร์ใช้เวลาวัดไม่เกิน 15 มิลลิวินาที

  if (Wire.requestFrom(SHT30_ADDR, (uint8_t)6) != 6) return false;
  uint8_t b[6];
  for (int i = 0; i < 6; i++) b[i] = Wire.read();

  uint16_t rawT = (b[0] << 8) | b[1];   // ไบต์ 0–1 = อุณหภูมิ (ไบต์ 2 คือรหัสตรวจ CRC)
  uint16_t rawH = (b[3] << 8) | b[4];   // ไบต์ 3–4 = ความชื้น
  tempC = -45 + 175.0 * rawT / 65535;   // สูตรจากเอกสารของผู้ผลิต
  humid = 100.0 * rawH / 65535;
  return true;
}

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

void loop() {
```

```
  float t, h;
  if (readSHT30(t, h)) {
    Serial.print("อุณหภูมิ °C = ");
    Serial.println(t, 2);
    Serial.print("ความชื้น %RH = ");
    Serial.println(h, 2);
  } else {
    Serial.println("อ่าน SHT30 ไม่สำเร็จ — ตรวจสาย Grove");
  }
  delay(1000);
}
```

🔍 บรรทัดสำคัญ

โค้ด	ความหมาย
`Wire.beginTransmission(SHT30_ADDR);`	เริ่มคุยกับอุปกรณ์ที่ที่อยู่ 0x44
`if (Wire.endTransmission() != 0) return false;`	ส่งคำสั่งจริง — ถ้าไม่มีใครตอบ (สายหลุด/ที่อยู่ผิด) ให้เลิกและบอกว่าอ่านไม่สำเร็จ
`tempC = -45 + 175.0 * rawT / 65535;`	แปลงเลขดิบเป็น °C · ต้องเขียน **175.0** ไม่ใช่ 175 ไม่งั้นคอมพิวเตอร์หารแบบจำนวนเต็มแล้วเศษหาย
`bool readSHT30(float &tempC, float &humid)`	เครื่องหมาย `&` แปลว่า “เขียนผลกลับใส่ตัวแปรของคนเรียก” — ฟังก์ชันเดียวจึงคืนได้สองค่า

> 🔮 **ลองทำนายก่อนอัปโหลด** — ถ้าเป่าลมหายใจใส่เซนเซอร์ 3 วินาที ค่าไหนจะเปลี่ยน **เร็วกว่า** กัน — อุณหภูมิ หรือ ความชื้น?
>
> คำทำนายของฉัน: _____

▶️ อัปโหลดลงบอร์ด

กดปุ่ม → **Upload** (ลูกศรขวา มุมซ้ายบน) รอจนขึ้น **Done uploading** → เปิด **Serial Monitor** (ไอคอนแว่นขยายมุมขวาบน) ตั้ง **115200 baud**

👀 ผลที่ควรเห็น

- ทุก 1 วินาที Serial Monitor ขึ้นสองบรรทัด เช่น `อุณหภูมิ °C = 26.51` และ `ความชื้น %RH = 67.28`
- เป่าลมหายใจ — ความชื้นพุ่งขึ้นทันที อุณหภูมิค่อย ๆ ขึ้นตามทีหลัง
- ถ้าขึ้น `อ่าน SHT30 ไม่สำเร็จ` — ตรวจสาย Grove และลองโปรแกรมสแกน I2C (ภาคผนวก ก)

📝 บันทึกผลของฉัน

สถานการณ์	อุณหภูมิ (°C)	ความชื้น (%RH)
ในห้องปกติ		
หลังเป่าลมหายใจ 3 วินาที		
หลังกำเซนเซอร์ไว้ 30 วินาที		
ข้างหน้าต่าง / นอกห้อง		

> 💡 **รู้ไหม? Serial Plotter** — เมนู Tools → Serial Plotter จะวาดค่าที่พิมพ์ออกมาเป็นกราฟให้ทันที ลองเป่าลมใส่เซนเซอร์แล้วดูเส้นความชื้นพุ่งขึ้น (ถ้าอยากได้กราฟสวย ให้พิมพ์แค่ตัวเลขบรรทัดละค่า ไม่มีข้อความนำหน้า)

🚀 ท้าทายตัวเอง

⭐ เปลี่ยน `delay` เป็น 5000 แล้วเทียบกับ 1000 — แบบไหนเหมาะกับห้องเรียน? เพราะอะไร?

⭐⭐ แก้ให้พิมพ์บรรทัดเดียวแบบ `26.5,67.3` แล้วเปิด Serial Plotter ดูกราฟสองเส้น

⭐⭐⭐ คำนวณ **ดัชนีความร้อน** อย่างง่าย หรือ **จุดน้ำค้าง** (dew point) จาก t และ h แล้วพิมพ์ออกมา — ค้นสูตรเอง

LAB 4 แสดงอุณหภูมิบนจอ OLED

✅ คอมไพล์ผ่าน | ยังไม่ได้อัปโหลดลงบอร์ดจริง | เวลาโดยประมาณ 15 นาที · ไฟล์ `Lab04_OLED.ino`

🌏 **ใช้ที่ไหนในชีวิตจริง?** หน้าจอนาฬิกาอัจฉริยะ · เครื่องวัดไข้ · จอบนไมโครเวฟ — อุปกรณ์ที่ต้องบอกค่าได้เองโดยไม่ต้องต่อคอม

🎯 เป้าหมาย

- รวมสองอย่างที่รู้แล้ว: อ่าน SHT30 + แสดงบนจอ OLED
- เขียนฟังก์ชัน `showOnOled()` ไว้ใช้ซ้ำ
- รู้ว่าโปรแกรมถูกเก็บในบอร์ด ใช้ได้โดยไม่ต้องมีคอม

🧰 อุปกรณ์

บอร์ด + SHT30 + จอ OLED (ต่อผ่าน Grove Hub บนสาย I2C เดียวกัน)

💭 **แนวคิด** — จอ OLED กับ SHT30 **ใช้สาย I2C คู่เดียวกัน** (ต่อผ่าน Grove Hub) เพราะแต่ละตัวมีที่อยู่ไม่ซ้ำ (0x3C กับ 0x44) เหมือนบ้านหลายหลังในซอยเดียวกันที่มีเลขที่บ้านต่างกัน บุรุษไปรษณีย์จึงส่งจดหมายถูกหลัง

🧩 เทียบกับบล็อก MakerFlow

บล็อกในฉบับ MakerFlow	เขียนเป็นโค้ด Arduino
🌡️ SHT30	`readSHT30(t, h)` (ฟังก์ชันเดิมจาก Lab 3)
📺 จอ OLED (Temp C · ขนาด 3)	`showOnOled("Temp C", t, 1)`
🖥️ แสดงค่าบนหน้าจอ	`Serial.println(t, 2)`

⌨️ พิมพ์โค้ดนี้ (หรือเปิดไฟล์ `Lab04_OLED.ino`)

```
// Lab 4 — แสดงอุณหภูมิบนจอ OLED (SHT30 + จอ SH1106 บนสาย I2C เดียวกัน)
#include <Wire.h>
#include <Adafruit_SH110X.h>

Adafruit_SH1106G oled(128, 64, &Wire, -1);
const uint8_t SHT30_ADDR = 0x44;

bool readSHT30(float &tempC, float &humid) {
  Wire.beginTransmission(SHT30_ADDR);
  Wire.write(0x24); Wire.write(0x00);
  if (Wire.endTransmission() != 0) return false;
  delay(20);
  if (Wire.requestFrom(SHT30_ADDR, (uint8_t)6) != 6) return false;
  uint8_t b[6];
  for (int i = 0; i < 6; i++) b[i] = Wire.read();
  tempC = -45 + 175.0 * ((b[0] << 8) | b[1]) / 65535;
  humid = 100.0 * ((b[3] << 8) | b[4]) / 65535;
  return true;
}

// วาดป้ายกำกับเล็ก ๆ บรรทัดบน + ตัวเลขใหญ่ข้างล่าง (ใช้ซ้ำได้ทุก Lab)
void showOnOled(const char *label, float value, int decimals) {
  oled.clearDisplay();
  oled.setTextColor(SH110X_WHITE);
  oled.setTextSize(1);
  oled.setCursor(0, 0);
  oled.print(label);
  oled.setTextSize(3);
  oled.setCursor(0, 20);
  oled.print(value, decimals);
  oled.display();
}

void setup() {
```

```
  Serial.begin(115200);
  Wire.begin();
  oled.begin(0x3C, true);
}

void loop() {
  float t, h;
  if (readSHT30(t, h)) {
    showOnOled("Temp C", t, 1);
    Serial.print("อุณหภูมิ °C = ");
    Serial.println(t, 2);
  }
  delay(1000);
}
```

🔍 บรรทัดสำคัญ

โค้ด	ความหมาย
`void showOnOled(const char *label, float value, int decimals)`	ฟังก์ชันของเราเอง: รับป้าย ตัวเลข และจำนวนทศนิยม แล้ววาดให้ทั้งหน้า
`oled.setTextSize(3);`	ขนาดตัวอักษร 1 = สูง 8 จุด · 3 = สูง 24 จุด (จอสูงทั้งหมด 64 จุด)
`oled.print(value, decimals);`	พิมพ์ตัวเลขตามจำนวนทศนิยมที่ขอ เช่น 26.5

> 🔮 **ลองทำนายก่อนอัปโหลด** — ถ้าถอดสาย USB ออกจากคอม แล้วเสียบกับพาวเวอร์แบงก์ — จอจะยังแสดงอุณหภูมิไหม? เพราะอะไร?
>
> คำทำนายของฉัน:
> ---

▶️ อัปโหลดลงบอร์ด

กดปุ่ม → **Upload** (ลูกศรขวา มุมซ้ายบน) รอจนขึ้น **Done uploading** → เปิด **Serial Monitor** (ไอคอนแว่นขยายมุมขวาบน) ตั้ง **115200 baud**

👀 ผลที่ควรเห็น

- จอบนบอร์ดแสดง **Temp C** และตัวเลขอุณหภูมิ อัปเดตทุกวินาที
- ถอดสายจากคอมไปเสียบพาวเวอร์แบงก์ จอยังแสดงค่าต่อ

📝 บันทึกผลของฉัน

ขนาดตัวอักษร	อ่านออกจากระยะกี่เมตร
1	
2	
3	
4	

> 💡 **รู้ไหม? อัปโหลดแล้วโปรแกรมไปอยู่ไหน?** Arduino IDE แปลงโค้ด (คอมไพล์) เป็นภาษาเครื่อง แล้วเขียนลง **หน่วยความจำแฟลช** ในบอร์ด ซึ่งไม่ลบเมื่อไฟดับ (แบบเดียวกับแฟลชไดรฟ์) · เปิดเครื่องเมื่อไร บอร์ดก็เริ่ม `setup()` ใหม่เอง

🚀 ท้าทายตัวเอง

⭐ ให้จอแสดงความชื้นแทน และแก้ป้ายเป็น `Humid %`

⭐⭐ แสดง **ทั้งสองค่า** บนจอเดียว (อุณหภูมิบรรทัดบน ความชื้นบรรทัดล่าง ขนาด 2)

⭐⭐⭐ แปลง 0x44 และ 0x3C เป็นเลขฐานสิบ แล้วลองเปลี่ยน `oled.begin(0x3D, true)` — เกิดอะไรขึ้น? ทำไม?

LAB 5 คำนวณ — แปลงองศาเซลเซียสเป็นฟาเรนไฮต์

✅ คอมไพล์ผ่าน | ยังไม่ได้อัปโหลดลงบอร์ดจริง | เวลาโดยประมาณ 20 นาที · ไฟล์ `Lab05_Fahrenheit.ino`

🌏 **ใช้ที่ไหนในชีวิตจริง?** แอปพยากรณ์อากาศที่สลับ °C/°F · เครื่องชั่งที่แปลงกิโลกรัมเป็นปอนด์ · ตู้น้ำมันที่คำนวณราคาจากลิตร

🎯 เป้าหมาย

- ใช้ตัวดำเนินการคณิตศาสตร์ `* + - /` กับตัวแปรทศนิยม (`float`)
- แยกการคำนวณเป็นขั้น ๆ เพื่อตรวจทีละขั้น
- เทียบคำตอบในกระดาษกับค่าจากบอร์ด

🧰 อุปกรณ์

บอร์ด + SHT30 + จอ OLED

💭 **แนวคิด** — สูตร °F = °C × 1.8 + 32 เขียนเป็นโค้ดได้บรรทัดเดียว แต่เราแยกเป็น `step1` และ `f` เพื่อพิมพ์ทุกขั้นออกมาดู — เวลาผลผิดจะรู้ทันทีว่าผิดที่ขั้นไหน (นักเขียนโปรแกรมมืออาชีพก็ทำแบบนี้)

🧩 เทียบกับบล็อก MakerFlow

บล็อกในฉบับ MakerFlow	เขียนเป็นโค้ด Arduino
🧮 คำนวณ ×1.8	`float step1 = c * 1.8;`
🧮 คำนวณ +32	`float f = step1 + 32;`
📺 จอ OLED (Temp F)	`showOnOled("Temp F", f, 1)`

⌨️ พิมพ์โค้ดนี้ (หรือเปิดไฟล์ `Lab05_Fahrenheit.ino`)

```
// Lab 5 — คำนวณ: แปลงองศาเซลเซียสเป็นฟาเรนไฮต์  (°F = °C × 1.8 + 32)
#include <Wire.h>
#include <Adafruit_SH110X.h>

Adafruit_SH1106G oled(128, 64, &Wire, -1);
const uint8_t SHT30_ADDR = 0x44;

bool readSHT30(float &tempC, float &humid) {
  Wire.beginTransmission(SHT30_ADDR);
  Wire.write(0x24); Wire.write(0x00);
  if (Wire.endTransmission() != 0) return false;
  delay(20);
  if (Wire.requestFrom(SHT30_ADDR, (uint8_t)6) != 6) return false;
  uint8_t b[6];
  for (int i = 0; i < 6; i++) b[i] = Wire.read();
  tempC = -45 + 175.0 * ((b[0] << 8) | b[1]) / 65535;
  humid = 100.0 * ((b[3] << 8) | b[4]) / 65535;
  return true;
}

void showOnOled(const char *label, float value, int decimals) {
  oled.clearDisplay();
  oled.setTextColor(SH110X_WHITE);
  oled.setTextSize(1);
  oled.setCursor(0, 0);
  oled.print(label);
  oled.setTextSize(3);
  oled.setCursor(0, 20);
  oled.print(value, decimals);
  oled.display();
}
```

```
void setup() {
  Serial.begin(115200);
  Wire.begin();
  oled.begin(0x3C, true);
}

void loop() {
  float c, h;
  if (readSHT30(c, h)) {
    float step1 = c * 1.8;          // ขั้นที่ 1: คูณ 1.8
    float f = step1 + 32;           // ขั้นที่ 2: บวก 32

    showOnOled("Temp F", f, 1);
    Serial.print(c, 2);   Serial.print(" °C  × 1.8 = ");
    Serial.print(step1, 2); Serial.print("  + 32 = ");
    Serial.print(f, 2);   Serial.println(" °F");
  }
  delay(1000);
}
```

🔍 บรรทัดสำคัญ

โค้ด	ความหมาย
`float step1 = c * 1.8;`	`float` = ตัวเลขมีทศนิยม · `int` = จำนวนเต็ม (ถ้าใช้ int ทศนิยมหาย)
`Serial.print(step1, 2);`	พิมพ์ทศนิยม 2 ตำแหน่ง

> 🔮 **ลองทำนายก่อนอัปโหลด** — ถ้าห้องร้อน 25 °C จะเท่ากับกี่ °F? คิดในกระดาษก่อน แล้วค่อยเทียบกับจอ
>
> คำทำนายของฉัน: _____

▶️ อัปโหลดลงบอร์ด

กดปุ่ม → **Upload** (ลูกศรขวา มุมซ้ายบน) รอจนขึ้น **Done uploading** → เปิด **Serial Monitor** (ไอคอนแว่นขยายมุมขวาบน) ตั้ง **115200 baud**

👀 ผลที่ควรเห็น

- Serial Monitor ขึ้นทุกขั้น เช่น `26.54 °C × 1.8 = 47.77 + 32 = 79.77 °F`
- จอ OLED แสดง **Temp F** กับตัวเลขประมาณ 79.8

📝 บันทึกผลของฉัน

°C จากเซนเซอร์	คิดเองได้ °F	จอแสดง °F	ตรงกันไหม

> 💡 **รู้ไหม? ลำดับการคำนวณ** ในโค้ดเหมือนคณิตศาสตร์: คูณ/หาร ก่อน บวก/ลบ · `c * 1.8 + 32` จึงถูก แต่ `(c + 32) * 1.8` ผิด — วงเล็บเปลี่ยนลำดับได้

🚀 ท้าทายตัวเอง

⭐ แปลงเป็นเคลวิน (K = °C + 273.15) แสดงบนจอ

⭐⭐ แสดงเป็นจำนวนเต็มด้วย `round(f)` — ต่างจาก `(int)f` อย่างไร? ลองที่ 79.8

⭐⭐⭐ ลองเขียนผิดเป็น `(c + 32) * 1.8` — ผิดไปกี่ °F? ผิดเท่าเดิมทุกอุณหภูมิไหม? พิสูจน์ด้วยพีชคณิต

LAB 6 ตัดสินใจ — ไฟเตือนเมื่อร้อนเกินกำหนด

✅ คอมไพล์ผ่าน | ยังไม่ได้อัปโหลดลงบอร์ดจริง | เวลาโดยประมาณ 20 นาที · ไฟล์ `Lab06_HeatAlarm.ino`

🌏 **ใช้ที่ไหนในชีวิตจริง?** ไฟเตือนความร้อนเครื่องยนต์บนหน้าปัดรถ · ตู้เย็นที่ร้องเตือนเมื่อลืมปิดประตู · สัญญาณเตือนไฟไหม้

🎯 เป้าหมาย

- ใช้ `if … else` ให้บอร์ด **ตัดสินใจ**
- รู้จักชนิดข้อมูล `bool` (true / false)
- ตั้งเกณฑ์เป็นค่าคงที่ไว้บนสุด แก้ง่าย

🧰 อุปกรณ์

บอร์ด + SHT30 (ใช้ไฟ LED บนบอร์ดเป็นไฟเตือน)

💭 **แนวคิด** — `t > HOT_LIMIT` คือคำถามที่คำตอบมีแค่ **ใช่ / ไม่ใช่** (true / false) · เก็บคำตอบไว้ในตัวแปร `tooHot` แล้วใช้ `if` เลือกทาง: ใช่ → เปิดไฟ · ไม่ใช่ → ปิดไฟ เหมือนทางแยกบนถนนที่มีป้ายบอก

🧩 เทียบกับบล็อก MakerFlow

บล็อกในฉบับ MakerFlow	เขียนเป็นโค้ด Arduino
⚖️ เปรียบเทียบ (มากกว่า 30)	`bool tooHot = (t > HOT_LIMIT);`
💡 สั่งขาดิจิทัล (LED กลับค่า)	`if (tooHot) digitalWrite(LED_BUILTIN, LOW); else … HIGH`

⌨️ พิมพ์โค้ดนี้ (หรือเปิดไฟล์ `Lab06_HeatAlarm.ino`)

```
// Lab 6 — ตัดสินใจ: ไฟเตือนเมื่อร้อนเกินกำหนด
#include <Wire.h>

const uint8_t SHT30_ADDR = 0x44;
const float HOT_LIMIT = 30.0;      // เกณฑ์ "ร้อน" (องศาเซลเซียส)

bool readSHT30(float &tempC, float &humid) {
  Wire.beginTransmission(SHT30_ADDR);
  Wire.write(0x24); Wire.write(0x00);
  if (Wire.endTransmission() != 0) return false;
  delay(20);
  if (Wire.requestFrom(SHT30_ADDR, (uint8_t)6) != 6) return false;
  uint8_t b[6];
  for (int i = 0; i < 6; i++) b[i] = Wire.read();
  tempC = -45 + 175.0 * ((b[0] << 8) | b[1]) / 65535;
  humid = 100.0 * ((b[3] << 8) | b[4]) / 65535;
  return true;
}

void setup() {
  Serial.begin(115200);
  Wire.begin();
  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, HIGH);   // เริ่มต้นไฟดับ (active-low)
}

void loop() {
  float t, h;
  if (readSHT30(t, h)) {
    bool tooHot = (t > HOT_LIMIT);   // ผลเปรียบเทียบเป็น true หรือ false

    if (tooHot) {
      digitalWrite(LED_BUILTIN, LOW);    // ร้อน → ไฟติด
    } else {
      digitalWrite(LED_BUILTIN, HIGH);   // ไม่ร้อน → ไฟดับ
    }
```

```
    Serial.print("อุณหภูมิ °C = ");
    Serial.print(t, 2);
    Serial.print("   ร้อนเกินไหม = ");
    Serial.println(tooHot ? "true" : "false");
  }
  delay(1000);
}
```

🔍 บรรทัดสำคัญ

โค้ด	ความหมาย
`const float HOT_LIMIT = 30.0;`	`const` = ค่าคงที่ แก้ที่บรรทัดนี้ที่เดียว ทั้งโปรแกรมเปลี่ยนตาม
`Serial.println(tooHot ? "true" : "false");`	`เงื่อนไข ? ก : ข` คือ if แบบย่อบรรทัดเดียว — จริงได้ ก เท็จได้ ข

> 👩‍🏫 **สำหรับครู:** ให้นักเรียนอ่านอุณหภูมิห้องจาก Lab 3 ก่อน แล้วตั้งเกณฑ์ “สูงกว่าอุณหภูมิห้อง 1 องศา” จะเห็นไฟติดเมื่อกำเซนเซอร์ภายในไม่กี่สิบวินาที — ห้องจริงไม่ค่อยร้อนถึง 30 °C · ฉบับ Arduino ต้อง **อัปโหลดใหม่ทุกครั้งที่แก้เกณฑ์** (ราว 20–30 วินาที) ต่างจาก MakerFlow ที่ส่งได้ทันที

> 🔮 **ลองทำนายก่อนอัปโหลด** — อุณหภูมิห้องตอนนี้กี่องศา (ดูจาก Lab 3)? — ตอนเริ่ม ไฟจะ **ติด** หรือ **ดับ**?
>
> คำทำนายของฉัน: _____

▶️ อัปโหลดลงบอร์ด

กดปุ่ม → **Upload** (ลูกศรขวา มุมซ้ายบน) รอจนขึ้น **Done uploading** → เปิด **Serial Monitor** (ไอคอนแว่นขยายมุมขวาบน) ตั้ง **115200 baud**

👀 ผลที่ควรเห็น

- ห้องปกติ (ทดสอบจริง 26.5 °C): `ร้อนเกินไหม = false` ไฟดับ
- แก้ `HOT_LIMIT` ให้ต่ำกว่าอุณหภูมิห้อง (เช่น 26) แล้วอัปโหลดใหม่: ได้ true และไฟติดค้าง
- ตั้งเกณฑ์สูงกว่าห้อง 1 องศา แล้วกำเซนเซอร์ไว้ — ไฟติดภายในไม่กี่สิบวินาที

📝 บันทึกผลของฉัน

เกณฑ์ที่ตั้ง (°C)	อุณหภูมิที่วัดได้	ร้อนเกินไหม	ไฟ LED
30			
อุณหภูมิห้อง + 1 = ...			
อุณหภูมิห้อง - 1 = ...			

> 💡 **รู้ไหม? = กับ == ต่างกัน!** `x = 5` คือ *ใส่* 5 ลงกล่อง x · `x == 5` คือ *ถาม* ว่า x เท่ากับ 5 ไหม — เขียน `if (t = 30)` ผิดเป็นข้อผิดพลาดยอดฮิตอันดับหนึ่งของมือใหม่ (อุณหภูมิจะกลายเป็น 30 และเงื่อนไขเป็นจริงตลอด)

🚀 ท้าทายตัวเอง

⭐ ให้ไฟติดเมื่อ *เย็น* เกินไป (ห้องแอร์ต่ำกว่า 24 °C) — แก้เครื่องหมายตัวไหน?

⭐⭐ เตือนเมื่อความชื้นเกิน 80 %RH แทนอุณหภูมิ

⭐⭐⭐ ถ้าอุณหภูมิห้องวนอยู่แถว 30.0 พอดี ไฟจะติด-ดับกะพริบ ออกแบบ **สองเกณฑ์** (เปิดที่ > 30.5 ปิดที่ < 29.5) — ต้องจำสถานะเดิมไว้ในตัวแปรอะไร?

LAB 7 รีเลย์ 2 ช่อง — พัดลมและเครื่องลดความชื้นอัตโนมัติ

✅ คอมไพล์ผ่าน | ยังไม่ได้อัปโหลดลงบอร์ดจริง | เวลาโดยประมาณ 25 นาที · ไฟล์ `Lab07_Relays.ino`

🌏 **ใช้ที่ไหนในชีวิตจริง?** ระบบรดน้ำต้นไม้อัตโนมัติ · พัดลมระบายอากาศในห้องน้ำ · ฟาร์มเห็ดที่คุมความชื้น

🎯 เป้าหมาย

- สั่งรีเลย์ 2 ช่องบนบอร์ดขยาย (ขา D0 และ D1)
- ใช้สองเงื่อนไขที่ทำงานแยกกัน
- เข้าใจว่ารีเลย์คือ “สวิตช์ที่บอร์ดกดได้เอง”

🧰 อุปกรณ์

บอร์ด + บอร์ดขยาย AI Expansion (รีเลย์ช่อง 1 และ 2) + SHT30

💭 **แนวคิด** — ขา D0/D1 ของบอร์ดจ่ายไฟได้นิดเดียว ขับพัดลมตรง ๆ ไม่ไหว บอร์ดขยายจึงมี **รีเลย์** — สวิตช์ไฟฟ้าที่บอร์ดสั่งด้วยไฟเล็ก ๆ แล้วมันไปตัด-ต่อวงจรไฟของอุปกรณ์อีกวงหนึ่ง · สองเงื่อนไขในโปรแกรมนี้ไม่เกี่ยวกัน: ร้อน → ช่อง 1 · ชื้น → ช่อง 2

🧩 เทียบกับบล็อก MakerFlow

บล็อกในฉบับ MakerFlow	เขียนเป็นโค้ด Arduino
⚖️ เปรียบเทียบ ร้อน > 30	`bool fanOn = (t > HOT_LIMIT);`
⚖️ เปรียบเทียบ ชื้น > 70	`bool dryerOn = (h > HUMID_LIMIT);`
🔌 รีเลย์ช่อง 1 / ช่อง 2	`digitalWrite(D0 / D1, HIGH)` = รีเลย์ทำงาน

⚠️ **ความปลอดภัย — ห้ามต่อไฟบ้าน 220 V กับรีเลย์ในห้องเรียนเด็ดขาด** ใช้เฉพาะอุปกรณ์ไฟต่ำ 5–12 V (พัดลม USB, หลอด LED) · ถอดสาย USB ทุกครั้งก่อนต่อหรือถอดสายที่ขั้วรีเลย์ · งานไฟบ้านให้ครูสาธิตด้วยปลั๊กรีเลย์สำเร็จรูปเท่านั้น

⌨️ พิมพ์โค้ดนี้ (หรือเปิดไฟล์ `Lab07_Relays.ino`)

```
// Lab 7 — รีเลย์ 2 ช่อง: พัดลมและเครื่องลดความชื้นอัตโนมัติ
// บอร์ดขยาย AI Expansion: รีเลย์ช่อง 1 = ขา D0 · ช่อง 2 = ขา D1 (HIGH = รีเลย์ทำงาน)
#include <Wire.h>

const uint8_t SHT30_ADDR = 0x44;
const int RELAY_FAN   = D0;        // ช่อง 1 — พัดลม
const int RELAY_DRYER = D1;        // ช่อง 2 — เครื่องลดความชื้น
const float HOT_LIMIT   = 30.0;    // °C
const float HUMID_LIMIT = 70.0;    // %RH

bool readSHT30(float &tempC, float &humid) {
  Wire.beginTransmission(SHT30_ADDR);
  Wire.write(0x24); Wire.write(0x00);
  if (Wire.endTransmission() != 0) return false;
  delay(20);
  if (Wire.requestFrom(SHT30_ADDR, (uint8_t)6) != 6) return false;
  uint8_t b[6];
  for (int i = 0; i < 6; i++) b[i] = Wire.read();
  tempC = -45 + 175.0 * ((b[0] << 8) | b[1]) / 65535;
  humid = 100.0 * ((b[3] << 8) | b[4]) / 65535;
  return true;
}

void setup() {
  Serial.begin(115200);
  Wire.begin();
  pinMode(RELAY_FAN, OUTPUT);
  pinMode(RELAY_DRYER, OUTPUT);
```

```
  digitalWrite(RELAY_FAN, LOW);      // เริ่มต้นปิดทั้งสองช่อง
  digitalWrite(RELAY_DRYER, LOW);
}

void loop() {
  float t, h;
  if (readSHT30(t, h)) {
    bool fanOn   = (t > HOT_LIMIT);
    bool dryerOn = (h > HUMID_LIMIT);

    digitalWrite(RELAY_FAN,   fanOn   ? HIGH : LOW);
    digitalWrite(RELAY_DRYER, dryerOn ? HIGH : LOW);

    Serial.print(t, 1); Serial.print(" °C  ");
    Serial.print(h, 1); Serial.print(" %RH   พัดลม = ");
    Serial.print(fanOn ? "true" : "false");
    Serial.print("   ลดความชื้น = ");
    Serial.println(dryerOn ? "true" : "false");
  }
  delay(1000);
}
```

🔍 บรรทัดสำคัญ

โค้ด	ความหมาย
`const int RELAY_FAN = D0;`	ตั้งชื่อขาให้มีความหมาย อ่านโค้ดแล้วรู้ทันทีว่าขานี้ต่ออะไร
`digitalWrite(RELAY_FAN, fanOn ? HIGH : LOW);`	รีเลย์ไม่กลับค่าเหมือนไฟ LED: HIGH = ทำงาน · LOW = หยุด

> 🔮 **ลองทำนายก่อนอัปโหลด** — ห้องตอนนี้ รีเลย์ช่องไหนจะทำงาน? ช่อง 1 (ร้อน > 30) · ช่อง 2 (ชื้น > 70) · ทั้งคู่ · ไม่มีเลย
>
> คำทำนายของฉัน: _____

▶️ อัปโหลดลงบอร์ด

กดปุ่ม → **Upload** (ลูกศรขวา มุมซ้ายบน) รอจนขึ้น **Done uploading** → เปิด **Serial Monitor** (ไอคอนแว่นขยายมุมขวาบน) ตั้ง **115200 baud**

👀 ผลที่ควรเห็น

- ห้องปกติ (ทดสอบจริงในฉบับ MakerFlow 26.5 °C, 67 %RH): ทั้งสองช่อง false รีเลย์ไม่ทำงาน
- ลดเกณฑ์เป็น 26 °C และ 60 %RH แล้วอัปโหลดใหม่: ทั้งสองช่องเป็น true ได้ยิน “แกร๊ก” และไฟสถานะบนแผงติด

📝 บันทึกผลของฉัน

เกณฑ์ร้อน	เกณฑ์ชื้น	อุณหภูมิ / ความชื้นที่วัดได้	ช่อง 1 แกร๊กไหม	ช่อง 2 แกร๊กไหม
30	70			
26	60			

> 💡 **รู้ไหม? ทำไมแค่ HIGH ก็พอ?** บนบอร์ดขยาย ขา D0/D1 ขับรีเลย์ผ่าน **ทรานซิสเตอร์ MOSFET** ที่ขยายกระแสให้ — ต่อรีเลย์เองต้องใส่วงจรนี้เอง

🚀 ท้าทายตัวเอง

⭐ เปลี่ยนเกณฑ์ชื้นเป็นค่า “อับชื้น” ที่ค้นไว้ใน Lab 3

⭐⭐ ให้ Serial Monitor พิมพ์เฉพาะตอนที่สถานะรีเลย์ **เปลี่ยน** (เช่น `พัดลม: เปิด`) ไม่ต้องพิมพ์ทุกวินาที

⭐⭐⭐ ให้พัดลมทำงานเฉพาะเมื่อ “ร้อน และ ชื้น” พร้อมกัน (ใบ้: `&&` — ดู Lab 11)

LAB 8 ไฟส่องสว่างอัตโนมัติด้วย BH1750

✅ คอมไพล์ผ่าน | ยังไม่ได้อัปโหลดลงบอร์ดจริง | เวลาโดยประมาณ 20 นาที · ไฟล์ `Lab08_AutoLight.ino`

🌏 **ใช้ที่ไหนในชีวิตจริง?** ไฟถนนที่ติดเองตอนค่ำ · จอมือถือที่ปรับความสว่างเอง · ไฟหน้ารถอัตโนมัติ

🎯 เป้าหมาย

- อ่านเซนเซอร์แสง BH1750 เป็นหน่วย lux
- ใช้ “น้อยกว่า” ตัดสินใจเปิดไฟอัตโนมัติ
- เห็นปัญหาไฟกะพริบตอนค่าอยู่แถวเกณฑ์พอดี

🧰 อุปกรณ์

บอร์ด + เซนเซอร์แสง BH1750 (ที่อยู่ 0x23) + จอ OLED + รีเลย์ช่อง 1

💭 **แนวคิด** – BH1750 ต่างจาก SHT30 ตรงที่สั่ง **ครั้งเดียว** ใน `setup()` ให้วัดต่อเนื่อง (คำสั่ง 0x10) หลังจากนั้นอ่านผลได้เรื่อย ๆ โดยไม่ต้องสั่งใหม่ · ค่าที่ได้หาร 1.2 จะเป็นหน่วย lux

🧩 เทียบกับบล็อก MakerFlow

บล็อกในฉบับ MakerFlow	เขียนเป็นโค้ด Arduino
🔆 BH1750 (0x23 · ละเอียด 1 lux)	`startBH1750()` ใน setup + `readLux()` ใน loop
⚖️ เปรียบเทียบ น้อยกว่า 100	`bool dark = (lux < DARK_LUX);`
🔌 รีเลย์ช่อง 1	`digitalWrite(RELAY_LIGHT, dark ? HIGH : LOW)`
📺 จอ OLED (Light lux · ขนาด 2)	`showOnOled("Light lux", lux, 1)`

⚠️ **ความปลอดภัย** – รีเลย์ในแล็บนี้ต่อได้เฉพาะหลอด LED / อุปกรณ์ไฟต่ำ 5–12 V – ห้ามต่อหลอดไฟบ้าน 220 V

⌨️ พิมพ์โค้ดนี้ (หรือเปิดไฟล์ `Lab08_AutoLight.ino`)

```
// Lab 8 – ไฟส่องสว่างอัตโนมัติด้วยเซนเซอร์แสง BH1750 (ที่อยู่ 0x23)
#include <Wire.h>
#include <Adafruit_SH110X.h>

Adafruit_SH1106G oled(128, 64, &Wire, -1);
const uint8_t BH1750_ADDR = 0x23;
const int RELAY_LIGHT = D0;        // รีเลย์ช่อง 1 – ไฟส่องสว่าง
const float DARK_LUX = 100;        // น้อยกว่านี้ถือว่ามืด

void startBH1750() {
  Wire.beginTransmission(BH1750_ADDR);
  Wire.write(0x10);                // โหมด "วัดต่อเนื่อง ละเอียด 1 lux"
  Wire.endTransmission();
}

// คืนค่าความสว่างเป็น lux · คืน -1 ถ้าอ่านไม่สำเร็จ
float readLux() {
  if (Wire.requestFrom(BH1750_ADDR, (uint8_t)2) != 2) return -1;
  uint8_t hi = Wire.read();        // อ่านไบต์สูงก่อน แล้วค่อยไบต์ต่ำ
  uint8_t lo = Wire.read();
  uint16_t raw = (hi << 8) | lo;
  return raw / 1.2;                // สูตรจากเอกสารของผู้ผลิต
}

void showOnOled(const char *label, float value, int decimals) {
  oled.clearDisplay();
  oled.setTextColor(SH110X_WHITE);
  oled.setTextSize(1);
  oled.setCursor(0, 0);
  oled.print(label);
```

```
  oled.setTextSize(2);
  oled.setCursor(0, 20);
  oled.print(value, decimals);
  oled.display();
}

void setup() {
  Serial.begin(115200);
  Wire.begin();
  oled.begin(0x3C, true);
  startBH1750();
  pinMode(RELAY_LIGHT, OUTPUT);
  digitalWrite(RELAY_LIGHT, LOW);
}

void loop() {
  float lux = readLux();
  if (lux >= 0) {
    bool dark = (lux < DARK_LUX);
    digitalWrite(RELAY_LIGHT, dark ? HIGH : LOW);
    showOnOled("Light lux", lux, 1);

    Serial.print("ความสว่าง = ");
    Serial.print(lux, 1);
    Serial.print(" lux   มืดไหม = ");
    Serial.println(dark ? "true" : "false");
  }
  delay(500);
}
```

🔍 บรรทัดสำคัญ

โค้ด	ความหมาย
`uint8_t hi = Wire.read(); uint8_t lo = Wire.read();`	อ่านไบต์สูงก่อน แล้วไบต์ต่ำ — **ต้องแยกเป็นสองบรรทัด** เพราะ C++ ไม่รับประกันลำดับถ้าเขียนรวมในสูตรเดียว
`return raw / 1.2;`	สูตรจากเอกสารผู้ผลิต BH1750
`if (lux >= 0)`	readLux() คืน -1 เมื่ออ่านพลาด — แสงติดลบเป็นไปไม่ได้ จึงใช้เป็นสัญญาณผิดพลาดได้

> 👩‍🏫 **สำหรับครู:** ถ้าไฟกะพริบถี่ ๆ ตอนแสงอยู่แถว 100 lux พอดี นั่นคือปัญหาจริงของระบบอัตโนมัติ — ชวนนักเรียนคิดวิธีแก้ (สองเกณฑ์ เปิดที่ < 80 ปิดที่ > 120) เชื่อมกับโจทย์ ⭐⭐⭐ ของ Lab 6

> 🔮 **ลองทำนายก่อนอัปโหลด** — ใต้โต๊ะเรียนมืดพอให้ไฟติดไหม (ต่ำกว่า 100 lux ไหม)? เดาก่อนแล้ววัดจริง
>
> คำทำนายของฉัน: _____

▶️ อัปโหลดลงบอร์ด

กดปุ่ม → **Upload** (ลูกศรขวา มุมซ้ายบน) รอจนขึ้น **Done uploading** → เปิด **Serial Monitor** (ไอคอนแว่นขยายมุมขวาบน) ตั้ง **115200 baud**

👀 ผลที่ควรเห็น

- ที่สว่าง: `มืดไหม = false` · จอ OLED แสดง **Light lux** และตัวเลขแสง
- เอามือปิดเซนเซอร์: ค่าลดต่ำกว่า 100 → true และรีเลย์ทำงาน (ได้ยิน “แกร๊ก”)
- ห้องเรียนเปิดไฟปกติมักได้ 200–500 lux · ห้องปิดไฟบางส่วนวัดจริงได้ ~23 lux

📝 บันทึกผลของฉัน

ที่ที่วัด	ค่าแสง (lux)	มืดไหม

กลางห้อง		
ใต้โต๊ะ		
เอามือปิดเซนเซอร์		
ข้างหน้าต่าง		
ส่องไฟฉายมือถือ		

> 💡 **รู้ไหม?** lux คือหน่วยความสว่างที่ตกกระทบพื้นผิว · ห้องเรียนควรได้ราว 300–500 lux · กลางแดดจ้าเกิน 50,000 lux · BH1750 วัดได้ถึง 65,535 lux จึงใช้วัดกลางแจ้งได้

🚀 ท้าทายตัวเอง

⭐ ปรับเกณฑ์ให้เหมาะกับห้องของเรา (ใช้ค่าจากตารางบันทึก)

⭐⭐ ใช้ `map()` และ `constrain()` แปลง 0–1000 lux เป็น 0–100 % แล้วแสดงบนจอ

⭐⭐⭐ แก้ไฟกะพริบด้วยสองเกณฑ์ (เปิด < 80 · ปิด > 120) แล้วทดสอบโดยค่อย ๆ เลื่อนมือบังเซนเซอร์

LAB 9 รดน้ำต้นไม้อัตโนมัติด้วยเซนเซอร์ความชื้นในดิน

✅ คอมไพล์ผ่าน | ยังไม่ได้อัปโหลดลงบอร์ดจริง | เวลาโดยประมาณ 30 นาที · ไฟล์ `Lab09a_SoilRaw.ino` · `Lab09b_SoilPump.ino`

🌏 **ใช้ที่ไหนในชีวิตจริง?** ระบบรดน้ำอัตโนมัติในแปลงผัก · กระถางอัจฉริยะที่เตือนเมื่อต้นไม้ขาดน้ำ · ฟาร์มอัจฉริยะ (smart farm) ที่ประหยัดน้ำ

🎯 เป้าหมาย

- อ่านสัญญาณ **แอนะล็อก** ด้วย `analogRead` (0–4095)
- **สอบเทียบ** (calibrate) เซนเซอร์ของเราเอง แล้วแปลงเป็นเปอร์เซ็นต์
- สั่งปั๊มน้ำผ่านรีเลย์ช่อง 2

🧰 อุปกรณ์

บอร์ด + เซนเซอร์ความชื้นในดินแบบแอนะล็อก (ขาสัญญาณ AOUT → D3 · VCC → 3V3 · GND → GND) + จอ OLED + รีเลย์ช่อง 2 (ปั๊มน้ำ USB 5 V)

💭 **แนวคิด** – เซนเซอร์ดินไม่ได้บอก “เปอร์เซ็นต์” มาให้ มันบอกแค่แรงดันไฟ ซึ่งบอร์ดแปลงเป็นเลข 0–4095 · เซนเซอร์แต่ละตัวให้เลขไม่เท่ากัน จึงต้องทำ **ขั้นเตรียม** (Lab09a) จดค่าตอนแห้งและตอนเปียกของตัวเราเองก่อน แล้วค่อยใส่ใน Lab09b เหมือนตั้งศูนย์ตาชั่งก่อนชั่งของ

🧩 เทียบกับบล็อก MakerFlow

บล็อกในฉบับ MakerFlow	เขียนเป็นโค้ด Arduino
📈 อ่านค่าแอนะล็อก (ขั้นเตรียม)	Lab09a: `analogRead(D3)`
🌱 ความชื้นในดิน (แห้ง 3000 · เปียก 1300)	Lab09b: ฟังก์ชัน `soilPercent(raw)`
⚖️ เปรียบเทียบ น้อยกว่า 40	`bool dry = (pct < DRY_LIMIT);`
🔌 รีเลย์ช่อง 2	`digitalWrite(RELAY_PUMP, dry ? HIGH : LOW)`

⚠️ **ความปลอดภัย** – ปั๊มน้ำใช้เฉพาะรุ่น USB / 5 V ผ่านรีเลย์ · วางบอร์ดให้ห่างจากน้ำ และเช็ดมือให้แห้งก่อนจับบอร์ด – ห้ามต่อปั๊มไฟบ้าน 220 V

⌨️ พิมพ์โค้ดนี้ (หรือเปิดไฟล์ `Lab09a_SoilRaw.ino`)

```
// Lab 9 ขั้นเตรียม – อ่านค่าดิบจากเซนเซอร์ความชื้นในดิน (ขาสัญญาณ AOUT → D3)
// จดค่าตอน "ถือในอากาศ/ดินแห้ง" และตอน "จุ่มน้ำ/ดินเปียก" ไว้ใช้ใน Lab09b

const int SOIL_PIN = D3;

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

void loop() {
  int raw = analogRead(SOIL_PIN);  // ได้ตัวเลข 0–4095 (ความละเอียด 12 บิต)
  Serial.print("ค่าดิบ = ");
  Serial.println(raw);
  delay(500);
}
```

⌨️ พิมพ์โค้ดนี้ (หรือเปิดไฟล์ `Lab09b_SoilPump.ino`)

```
// Lab 9 – รดน้ำต้นไม้อัตโนมัติด้วยเซนเซอร์ความชื้นในดิน
// ขาสัญญาณ AOUT → D3 · ปั๊มน้ำต่อรีเลย์ช่อง 2 (ขา D1)
#include <Wire.h>
#include <Adafruit_SH110X.h>
```

```
Adafruit_SH1106G oled(128, 64, &Wire, -1);
const int SOIL_PIN   = D3;
const int RELAY_PUMP = D1;
const float DRY_VALUE = 3000;       // ค่าดิบตอนแห้ง — แก้เป็นค่าที่วัดได้จาก Lab09a
const float WET_VALUE = 1300;       // ค่าดิบตอนเปียก — แก้เป็นค่าที่วัดได้จาก Lab09a
const float DRY_LIMIT = 40;         // ความชื้นต่ำกว่านี้ (%) = ดินแห้ง ต้องรดน้ำ

// แปลงค่าดิบเป็นเปอร์เซ็นต์: แห้ง = 0 % · เปียก = 100 %
float soilPercent(int raw) {
  float pct = (DRY_VALUE - raw) / (DRY_VALUE - WET_VALUE) * 100.0;
  if (pct < 0)   pct = 0;
  if (pct > 100) pct = 100;
  return pct;
}

void showOnOled(const char *label, float value, int decimals) {
  oled.clearDisplay();
  oled.setTextColor(SH110X_WHITE);
  oled.setTextSize(1);
  oled.setCursor(0, 0);
  oled.print(label);
  oled.setTextSize(2);
  oled.setCursor(0, 20);
  oled.print(value, decimals);
  oled.display();
}

void setup() {
  Serial.begin(115200);
  Wire.begin();
  oled.begin(0x3C, true);
  pinMode(RELAY_PUMP, OUTPUT);
  digitalWrite(RELAY_PUMP, LOW);
}

void loop() {
  int raw = analogRead(SOIL_PIN);
  float pct = soilPercent(raw);
  bool dry = (pct < DRY_LIMIT);

  digitalWrite(RELAY_PUMP, dry ? HIGH : LOW);
  showOnOled("Soil %", pct, 0);

  Serial.print("ค่าดิบ = "); Serial.print(raw);
  Serial.print("   ความชื้น = "); Serial.print(pct, 0);
  Serial.print(" %   ดินแห้งไหม = ");
  Serial.println(dry ? "true" : "false");
  delay(1000);
}
```

🔍 บรรทัดสำคัญ

โค้ด	ความหมาย
`int raw = analogRead(SOIL_PIN);`	อ่านแรงดันที่ขา D3 เป็นเลข 0 (0 V) ถึง 4095 (3.3 V)
`float pct = (DRY_VALUE - raw) / (DRY_VALUE - WET_VALUE) * 100.0;`	เทียบบัญญัติไตรยางศ์: ค่าดิบ = DRY → 0 % · ค่าดิบ = WET → 100 % (ดินยิ่งเปียก ค่าดิบยิ่ง **น้อย**)
`if (pct < 0) pct = 0;`	กันค่าเกินช่วง เช่น แห้งกว่าตอนสอบเทียบจะได้ติดลบ

> 👩‍🏫 **สำหรับครู:** ขั้นเตรียมคือหัวใจของแล็บนี้ — ให้นักเรียนเห็นว่าเซนเซอร์ 2 ตัวรุ่นเดียวกันให้ค่าดิบไม่เท่ากัน (เทียบตารางข้ามกลุ่ม) · ถ้าไม่มีปั๊ม ใช้หลอด LED 5 V ที่รีเลย์ช่อง 2 แทนได้ · ⚠ Lab นี้ยังไม่ได้ลองกับเซนเซอร์ดินตัวจริง ครูควรลอง 1 ชุดก่อนสอน

🔮 **ลองทำนายก่อนอัปโหลด** — ถ้าจุ่มเซนเซอร์ในน้ำเปล่า กับปักในดินที่เพิ่งรดน้ำ อันไหนได้เปอร์เซ็นต์สูงกว่า? แล้วทั้งสองจะเกิน 40 % ไหม?

คำทำนายของฉัน:

▶️ อัปโหลดลงบอร์ด

กดปุ่ม → **Upload** (ลูกศรขวา มุมซ้ายบน) รอจนขึ้น **Done uploading** → เปิด **Serial Monitor** (ไอคอนแว่นขยายมุมขวาบน) ตั้ง **115200 baud**

👀 ผลที่ควรเห็น

- ขั้นเตรียม (Lab09a): ค่าดิบตอนอยู่ในอากาศสูงกว่าตอนจุ่มน้ำชัดเจน (ตัวอย่างทั่วไป ~3000 กับ ~1300 แต่ของแต่ละตัวต่างกัน)
- Lab09b ถือเซนเซอร์ในอากาศ: จอขึ้น **Soil %** ใกล้ 0 · `ดินแห้งไหม = true` · รีเลย์ช่อง 2 ทำงาน
- ปักในดินที่รดน้ำแล้ว: ค่าเกิน 40 % → false รีเลย์ปิด

📝 บันทึกผลของฉัน

ขั้นเตรียม / สถานการณ์	ค่าดิบ (0–4095)	ความชื้น (%)	ปั๊มทำงานไหม
ถือไว้ในอากาศ (= แห้ง)		—	—
จุ่มน้ำถึงขีด (= เปียก)		—	—
ดินแห้งในกระถาง			
ดินหลังรดน้ำ 1 แก้ว			
ดินหลังรดน้ำ 5 นาที			

💡 **รู้ไหม? ดิจิทัล vs แอนะล็อก** — `digitalRead` ตอบได้แค่ 0 หรือ 1 (มีไฟ/ไม่มีไฟ) ส่วน `analogRead` ตอบได้ 4096 ระดับ เหมือนสวิตช์ไฟธรรมดากับสวิตช์หรี่ไฟ · ESP32 วัดแอนะล็อกได้ดีในช่วงกลาง ๆ แต่ปลายใกล้ 0 V และ 3.3 V จะเพี้ยนเล็กน้อย

🚀 ท้าทายตัวเอง

⭐ ปรับเกณฑ์ 40 % ให้เหมาะกับต้นไม้ของเรา (กระบองเพชรชอบดินแห้ง ผักบุ้งชอบดินชื้น)

⭐⭐ อ่านค่า 10 ครั้งแล้ว **หาค่าเฉลี่ย** ก่อนแปลงเป็น % ให้ตัวเลขนิ่งขึ้น (ใช้ `for`)

⭐⭐⭐ ประหยัดน้ำ: ให้ปั๊มทำงาน **ครั้งละไม่เกิน 3 วินาที** แล้วพัก 30 วินาทีให้น้ำซึม ก่อนวัดใหม่ — กันน้ำล้นกระถาง

LAB 10 สัญญาณเตือนการเอียงด้วย MPU6050

✅ คอมไพล์ผ่าน | ยังไม่ได้อัปโหลดลงบอร์ดจริง | เวลาโดยประมาณ 25 นาที · ไฟล์ `Lab10_Tilt.ino`

🌏 **ใช้ที่ไหนในชีวิตจริง?** จอมือถือที่หมุนตามมือ · จอยเกมที่เอียงเพื่อบังคับ · สายรัดข้อมือนับก้าว · ถุงลมนิรภัยในรถ

🎯 เป้าหมาย

- อ่านความเร่งแกน X จาก MPU6050 เป็นหน่วย g
- ใช้ **ค่าสัมบูรณ์** (`fabs`) ให้เอียงซ้ายหรือขวาก็เตือนเหมือนกัน
- อ่านเซนเซอร์ถี่ขึ้น (ทุก 200 มิลลิวินาที)

🧰 อุปกรณ์

บอร์ด + MPU6050 (ที่อยู่ 0x68) · ใช้ไฟ LED บนบอร์ดเป็นไฟเตือน

💭 **แนวคิด** – ตอนบอร์ดวางราบ แรงโน้มถ่วง 1 g ดึงลงแกน Z ทั้งหมด แกน X จึงใกล้ 0 · พอเอียง แรงโน้มถ่วงบางส่วน “ไหล” มาอยู่แกน X (เอียงขวาเป็นบวก เอียงซ้ายเป็นลบ) · ใช้ `fabs()` ตัดเครื่องหมายลบทิ้ง แล้วเทียบกับ 0.5 g ≈ เอียง 30°

🧩 เทียบกับบล็อก MakerFlow

บล็อกในฉบับ MakerFlow	เขียนเป็นโค้ด Arduino
🎯 MPU6050 (ความเร่ง g)	`startMPU()` ใน setup + `readAccelX(x)`
🧮 คำนวณ ค่าสัมบูรณ์	`fabs(x)`
⚖️ เปรียบเทียบ มากกว่า 0.5	`bool tilted = (fabs(x) > TILT_LIMIT);`
💡 สั่งขาดิจิทัล (LED กลับค่า)	`digitalWrite(LED_BUILTIN, tilted ? LOW : HIGH)`

⌨️ พิมพ์โค้ดนี้ (หรือเปิดไฟล์ `Lab10_Tilt.ino`)

```
// Lab 10 – สัญญาณเตือนการเอียงด้วย MPU6050 (ที่อยู่ 0x68)
#include <Wire.h>

const uint8_t MPU_ADDR = 0x68;
const float TILT_LIMIT = 0.5;      // |ความเร่งแกน X| มากกว่านี้ (หน่วย g) = เอียง

void startMPU() {
  Wire.beginTransmission(MPU_ADDR);
  Wire.write(0x6B);                // รีจิสเตอร์ PWR_MGMT_1
  Wire.write(0x00);                // เขียน 0 = ปลุกชิปให้ตื่น
  Wire.endTransmission();
}

// อ่านความเร่งแกน X หน่วย g · คืน false ถ้าอ่านไม่สำเร็จ
bool readAccelX(float &gx) {
  Wire.beginTransmission(MPU_ADDR);
  Wire.write(0x3B);                // เริ่มอ่านที่รีจิสเตอร์ ACCEL_XOUT_H
  if (Wire.endTransmission(false) != 0) return false;
  if (Wire.requestFrom(MPU_ADDR, (uint8_t)2) != 2) return false;
  uint8_t hi = Wire.read();        // อ่านไบต์สูงก่อน แล้วค่อยไบต์ต่ำ
  uint8_t lo = Wire.read();
  int16_t raw = (hi << 8) | lo;
  gx = raw / 16384.0;              // ช่วง ±2g: 16384 = 1 g
  return true;
}

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

```
  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, HIGH);   // ไฟดับ
}

void loop() {
  float x;
  if (readAccelX(x)) {
    bool tilted = (fabs(x) > TILT_LIMIT);   // fabs = ค่าสัมบูรณ์ (ตัดเครื่องหมายลบทิ้ง)
    digitalWrite(LED_BUILTIN, tilted ? LOW : HIGH);

    Serial.print("ความเร่งแกน X (g) = ");
    Serial.print(x, 2);
    Serial.print("   เอียงไหม = ");
    Serial.println(tilted ? "true" : "false");
  }
  delay(200);
}
```

🔍 บรรทัดสำคัญ

โค้ด	ความหมาย
`Wire.write(0x6B); Wire.write(0x00);`	MPU6050 เปิดเครื่องมาใน **โหมดหลับ** — ต้องเขียน 0 ลงรีจิสเตอร์ 0x6B เพื่อปลุกก่อน ไม่งั้นอ่านได้ 0 ตลอด
`int16_t raw = (hi << 8) | lo;`	`int16_t` = จำนวนเต็มมีเครื่องหมาย จึงได้ค่าลบตอนเอียงอีกทาง
`gx = raw / 16384.0;`	ช่วงวัด ±2 g: ค่าดิบ 16384 = 1 g

> 👩‍🏫 **สำหรับครู:** ครูควรลองกับเซนเซอร์จริง 1 ชุดก่อนเริ่มคาบ และดูทิศลูกศร X/Y ที่พิมพ์บนแผ่น MPU6050 เพื่อบอกนักเรียนว่าแกนไหนคือแกนไหน · ถ้าใช้ไลบรารี MPU6050 ทั่วไปที่ตรวจรหัสรุ่น (WHO_AM_I) จะ **ไม่รู้จักชิปในคิท** (อ่านได้ 0x74) — โค้ดในใบงานนี้ไม่ตรวจรหัสรุ่น จึงใช้ได้

> 🔮 **ลองทำนายก่อนอัปโหลด** — ถ้าเอียงบอร์ดไปด้านข้างจน **ตั้งฉาก 90°** แกน X จะได้ประมาณเท่าไร?
>
> คำทำนายของฉัน: _____

▶️ อัปโหลดลงบอร์ด

กดปุ่ม → **Upload** (ลูกศรขวา มุมซ้ายบน) รอจนขึ้น **Done uploading** → เปิด **Serial Monitor** (ไอคอนแว่นขยายมุมขวาบน) ตั้ง **115200 baud**

👀 ผลที่ควรเห็น

- วางราบ: แกน X ใกล้ 0 ไฟดับ
- เอียงเกินประมาณ 30° ไปทางใดทางหนึ่ง: ไฟ LED ติด (ทดสอบจริงในฉบับ MakerFlow: X ≈ 0.60 g → เอียงไหม = true)

📝 บันทึกผลของฉัน

ท่าวางบอร์ด	แกน X (g)	เอียงไหม	ไฟ LED
วางราบบนโต๊ะ			
เอียงซ้ายเล็กน้อย			
เอียงขวาประมาณ 45°			
ตั้งฉาก 90°			
คว่ำหน้าลง			

> 💡 **รู้ไหม? ทำไม 0.5 g ≈ 30°?** แกน X ได้แรงโน้มถ่วง = 1 g × sin(มุมเอียง) และ sin 30° = 0.5 พอดี · มือถือใช้เซนเซอร์แบบนี้หมุนจอตามแนวที่ถือ · โมดูลในคิทนี้เป็นชิปตระกูล MPU-6500 ซึ่งรีจิสเตอร์ความเร่งเหมือน MPU6050 ทุกอย่าง

🚀 ท้าทายตัวเอง

⭐ ทำให้เตือนที่ 45° แทน 30° — ต้องตั้งเกณฑ์เท่าไร?

⭐⭐ ทำ “เครื่องวัดระดับน้ำ” — ให้ไฟติดเมื่อบอร์ด *ราบ* (|X| < 0.05)

⭐⭐⭐ อ่านแกน Y ด้วย (รีจิสเตอร์ถัดไป 0x3D) แล้วเตือนเมื่อเอียง **แกนใดแกนหนึ่ง** — แถม: คำนวณมุมเป็นองศาด้วย `asin()`

LAB 11 โปรเจกต์รวม — สถานีตรวจอากาศห้องเรียน

✅ คอมไพล์ผ่าน | ยังไม่ได้อัปโหลดลงบอร์ดจริง | เวลาโดยประมาณ 40 นาที · ไฟล์ `Lab11_Station.ino`

🌏 **ใช้ที่ไหนในชีวิตจริง?** สถานีตรวจอากาศ · บ้านอัจฉริยะ (smart home) · โรงเรือนเกษตรอัจฉริยะที่เปิดพัดลมและพ่นหมอกเอง

🎯 เป้าหมาย

- รวมทุกอย่างเป็นโปรเจกต์เดียว: เซนเซอร์ · จอ · ตรรกะ · รีเลย์
- ใช้ `for` ทำซ้ำ และ `&&` (และ)
- จับเวลาด้วย `millis()` แทน `delay()`

🧰 อุปกรณ์

บอร์ด + SHT30 + จอ OLED + รีเลย์ช่อง 1

💭 **แนวคิด** — `setup()` กะพริบไฟทักทาย 3 ครั้ง (ทำครั้งเดียว) · `loop()` ดูนาฬิกา `millis()` ถ้ายังไม่ครบ 2 วินาที ก็ออกไปก่อน ครบแล้วจึงวัดและตัดสินใจ · พัดลมเปิดเมื่อ **ร้อน และ ชื้น** พร้อมกันเท่านั้น

🧩 เทียบกับบล็อก MakerFlow

บล็อกในฉบับ MakerFlow	เขียนเป็นโค้ด Arduino
🚀 เมื่อเริ่มทำงาน + 📋 ลำดับขั้นตอน	`greetingBlink()` ที่ใช้ `for` 3 รอบ ใน setup
🌡️ SHT30 อ่านทุก 2000	`if (millis() - lastRead < 2000) return;`
🔗 ตรรกะ และ	`bool fanOn = hot && humid;`
🔌 รีเลย์ช่อง 1	`digitalWrite(RELAY_FAN, fanOn ? HIGH : LOW)`

⚠️ **ความปลอดภัย** — รีเลย์ต่อได้เฉพาะพัดลม USB / อุปกรณ์ไฟต่ำ 5–12 V — ห้ามต่อไฟบ้าน 220 V

⌨️ พิมพ์โค้ดนี้ (หรือเปิดไฟล์ `Lab11_Station.ino`)

```
// Lab 11 — โปรเจกต์รวม: สถานีตรวจอากาศห้องเรียน
// เปิดเครื่อง → ไฟทักทาย 3 ครั้ง → วัดทุก 2 วินาที → เปิดพัดลมเมื่อ "ร้อน และ ชื้น"
#include <Wire.h>
#include <Adafruit_SH110X.h>

Adafruit_SH1106G oled(128, 64, &Wire, -1);
const uint8_t SHT30_ADDR = 0x44;
const int RELAY_FAN = D0;
const float HOT_LIMIT   = 30.0;
const float HUMID_LIMIT = 60.0;
const unsigned long READ_EVERY_MS = 2000;

unsigned long lastRead = 0;

bool readSHT30(float &tempC, float &humid) {
  Wire.beginTransmission(SHT30_ADDR);
  Wire.write(0x24); Wire.write(0x00);
  if (Wire.endTransmission() != 0) return false;
  delay(20);
  if (Wire.requestFrom(SHT30_ADDR, (uint8_t)6) != 6) return false;
  uint8_t b[6];
  for (int i = 0; i < 6; i++) b[i] = Wire.read();
  tempC = -45 + 175.0 * ((b[0] << 8) | b[1]) / 65535;
  humid = 100.0 * ((b[3] << 8) | b[4]) / 65535;
  return true;
}
```

```
void showOnOled(const char *label, float value, int decimals) {
  oled.clearDisplay();
  oled.setTextColor(SH110X_WHITE);
  oled.setTextSize(1);
  oled.setCursor(0, 0);
  oled.print(label);
  oled.setTextSize(3);
  oled.setCursor(0, 20);
  oled.print(value, decimals);
  oled.display();
}

void greetingBlink() {
  for (int i = 0; i < 3; i++) {      // ทำซ้ำ 3 รอบ
    digitalWrite(LED_BUILTIN, LOW);  // ติด
    delay(200);
    digitalWrite(LED_BUILTIN, HIGH); // ดับ
    delay(200);
  }
  Serial.println("ไฟทักทายเสร็จแล้ว = true");
}

void setup() {
  Serial.begin(115200);
  Wire.begin();
  oled.begin(0x3C, true);
  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, HIGH);
  pinMode(RELAY_FAN, OUTPUT);
  digitalWrite(RELAY_FAN, LOW);
  greetingBlink();                   // setup() ทำครั้งเดียวตอนเปิดเครื่อง
}

void loop() {
  // ใช้ millis() จับเวลาแทน delay(2000) — บอร์ดจึงว่างทำอย่างอื่นได้ระหว่างรอ
  if (millis() - lastRead < READ_EVERY_MS) return;
  lastRead = millis();

  float t, h;
  if (!readSHT30(t, h)) {
    Serial.println("อ่าน SHT30 ไม่สำเร็จ");
    return;
  }
  bool hot   = (t > HOT_LIMIT);
  bool humid = (h > HUMID_LIMIT);
  bool fanOn = hot && humid;         // && = "และ" ต้องจริงทั้งคู่

  digitalWrite(RELAY_FAN, fanOn ? HIGH : LOW);
  showOnOled("Temp C", t, 1);

  Serial.print("อุณหภูมิ °C = "); Serial.print(t, 1);
  Serial.print("   ความชื้น %RH = "); Serial.print(h, 1);
  Serial.print("   เปิดพัดลม = ");
  Serial.println(fanOn ? "true" : "false");
}
```

🔍 บรรทัดสำคัญ

โค้ด	ความหมาย
`for (int i = 0; i < 3; i++)`	ทำซ้ำ 3 รอบ: เริ่ม i = 0 · ทำตราบที่ i < 3 · จบรอบเพิ่ม i ทีละ 1
`if (millis() - lastRead < READ_EVERY_MS) return;`	`millis()` = มิลลิวินาทีนับจากเปิดเครื่อง · ยังไม่ถึงเวลา → ออกจาก loop รอบนี้ไปก่อน
`bool fanOn = hot && humid;`	`&&` = และ (จริงทั้งคู่) · `||` = หรือ (จริงอย่างน้อยหนึ่ง) · `!` = ไม่

👩‍🏫 **สำหรับครู:** ใช้ ⭐⭐⭐ เป็นงานกลุ่มท้ายคาบ ให้แต่ละกลุ่มนำเสนอ 2 นาที: แก้ปัญหาอะไร ใช้โค้ดส่วนไหนจาก Lab ก่อน ๆ เจอปัญหาอะไรระหว่างทำ

🔮 **ลองทำนายก่อนอัปโหลด** – ห้องตอนนี้ พัดลมจะเปิดไหม? ถ้าไม่เปิด ต้องแก้เกณฑ์ตัวไหนให้เปิด?
คำทำนายของฉัน:

▶️ อัปโหลดลงบอร์ด

กดปุ่ม → **Upload** (ลูกศรขวา มุมซ้ายบน) รอจนขึ้น **Done uploading** → เปิด **Serial Monitor** (ไอคอนแว่นขยายมุมขวาบน) ตั้ง **115200 baud**

👀 ผลที่ควรเห็น

- เปิดเครื่อง: ไฟ LED กะพริบ 3 ครั้ง แล้ว Serial Monitor ขึ้น `ไฟกักกายเสร็จแล้ว = true`
- ห้องปกติ (ทดสอบจริงในฉบับ MakerFlow 26.6 °C / 67 %RH): ชื้นแต่ไม่ร้อน → `เปิดพัดลม = false`
- ลด `HOT_LIMIT` เป็น 26: ผลเป็น true และรีเลย์ช่อง 1 ทำงาน

📝 บันทึกผลของฉัน

ร้อน > 30 ?	ชื้น > 60 ?	ผลของ “และ”	พัดลม
จริง	จริง		
จริง	เท็จ		
เท็จ	จริง		
เท็จ	เท็จ		

💡 **รู้ไหม? ทำไมไม่ใช้ delay(2000)?** ระหว่าง delay บอร์ด “หลับ” ทำอย่างอื่นไม่ได้เลย – ถ้าวันหนึ่งเพิ่มปุ่มกด ปุ่มจะไม่ตอบสนอง 2 วินาที · แบบ millis() บอร์ดวน loop หลายหมื่นรอบต่อวินาที พร้อมทำงานอื่นเสมอ · เฟิร์มแวร์ MakerFlow ก็ทำงานแบบนี้ทุกบล็อก (ข้อความทักทายจึงขึ้นก่อน Serial Monitor เปิดทันไม่ได้ – กดปุ่ม RESET บนบอร์ดเพื่อดูใหม่)

🚀 ท้าทายตัวเอง

⭐ เพิ่มรีเลย์ช่อง 2 ให้ทำงานเมื่อ “ร้อน หรือ ชื้น” (`||`) แล้วเติมตารางค่าความจริงของ “หรือ”

⭐⭐ ให้ไฟ LED กะพริบ **ตลอดเวลา** ทุก 1 วินาทีเป็นไฟบอกว่าระบบยังทำงาน โดยใช้ millis() ตัวที่สอง – ห้ามใช้ delay

⭐⭐⭐ ออกแบบเอง: เพิ่มแสง (Lab 8) ความชื้นในดิน (Lab 9) หรือการเอียง (Lab 10) เข้าไปในสถานีนี้ – คัดลอกฟังก์ชันจาก Lab เดิมมาใช้

ภาคผนวก ก — ปัญหาที่พบบ่อย

อาการ	วิธีแก้
อัปโหลดไม่ขึ้น: `No serial data received` / `Failed to connect` / ไม่มีพอร์ตให้เลือก	เข้า **โหมดดาวน์โหลด**: ถอดสาย USB → **กดปุ่ม B (BOOT) ค้าง** → เสียบสาย → ปล่อยปุ่ม → เลือกพอร์ตใหม่แล้วอัปโหลด · อัปโหลดเสร็จกดปุ่ม R (RESET) หนึ่งครั้ง · ถ้ายังไม่เห็นพอร์ต ลองสาย USB เส้นอื่น (บางเส้นจ่ายไฟได้อย่างเดียว ส่งข้อมูลไม่ได้)
Serial Monitor ว่าง หรือขึ้นตัวอักษรมั่ว	ตั้งความเร็วมุมขวาเป็น **115200 baud** ให้ตรงกับ `Serial.begin(115200)` · เมนู Tools → USB CDC On Boot ต้องเป็น **Enabled** · ข้อความที่พิมพ์ใน setup() อาจหายไปก่อนเปิด Monitor ทัน — กดปุ่ม RESET บนบอร์ด
คอมไพล์ไม่ผ่าน: `Adafruit_SH110X.h: No such file or directory`	ยังไม่ได้ติดตั้งไลบรารี — Library Manager ค้นหา **Adafruit SH110X** → Install → กด **Install All** (ติดตั้ง Adafruit GFX และ BusIO ที่ต้องใช้คู่กันด้วย)
คอมไพล์ไม่ผ่าน: `'D0' was not declared` / `LED_BUILTIN` ไม่รู้จัก	เลือกบอร์ดผิด — Tools → Board → esp32 → **XIAO_ESP32S3**
ขึ้น `อ่าน SHT30 ไม่สำเร็จ` / จอ OLED ว่าง	ตรวจสาย Grove ให้เสียบแน่นที่ขั้ว I2C · อัปโหลดโปรแกรมสแกน I2C (ด้านล่าง) ควรเห็น 0x3C 0x44 0x23 0x68 ตามที่ต่อไว้ · จอว่างแต่สแกนเจอ 0x3C — ตรวจว่าโค้ดมี `oled.display()`
จอ OLED มีเส้นขยะ 2 คอลัมน์ทางขวา / ภาพเลื่อน	โค้ดใช้คลาส SSD1306 กับจอ SH1106 — จอในชุดต้องใช้ `Adafruit_SH1106G`
อยากกลับไปใช้ MakerFlow	เปิด maker-hub.net/makerflow → หัวข้อ “ติดตั้งเฟิร์มแวร์ลงบอร์ด” → ทำตามขั้นตอน (ราว 1 นาที) — งาน .json ที่บันทึกไว้ไม่หาย

โปรแกรมสแกน I2C (ไฟล์ `I2C_Scan.ino`)

```
// โปรแกรมสแกน I2C — พิมพ์ที่อยู่ของทุกอุปกรณ์ที่ต่ออยู่
#include <Wire.h>
void setup() { Serial.begin(115200); Wire.begin(); }
void loop() {
  Serial.println("กำลังสแกน...");
  for (uint8_t a = 1; a < 127; a++) {
    Wire.beginTransmission(a);
    if (Wire.endTransmission() == 0) { Serial.print("พบอุปกรณ์ที่ 0x"); Serial.println(a, HEX); }
  }
  delay(3000);
}
```

ภาคผนวก ข — ไฟล์ตัวอย่าง

โฟลเดอร์ `sketches/` มีสเก็ตช์ 13 ไฟล์ (11 Lab · Lab 9 แยกขั้นเตรียมกับตัวจริง · โปรแกรมสแกน I2C) แต่ละไฟล์อยู่ในโฟลเดอร์ชื่อเดียวกันตามที่ Arduino IDE กำหนด — ดับเบิลคลิกไฟล์ .ino เพื่อเปิดได้ทันที · ครูใช้เป็นเฉลย หรือให้นักเรียนที่ตามไม่ทันเริ่มจากไฟล์แล้วแก้ต่อ

ภาคผนวก ค — เฉลย “ลองทำนาย” และ “ท้าทายตัวเอง” (สำหรับครู)

โจทย์	แนวคำตอบ
Lab 1 · ทำนาย	10 ครั้ง — ติด 0.5 + ดับ 0.5 = 1 วินาทีต่อรอบ
Lab 1 · ⭐	ติด-ดับ 5 รอบต่อวินาที ยังเห็นกะพริบชัด
Lab 1 · ⭐⭐	แยกเป็น `ON_MS = 100` กับ `OFF_MS = 900` แล้วใช้คนละตัวใน delay สองบรรทัด
Lab 1 · ⭐⭐⭐	เขียนฟังก์ชัน `flash(int ms, int times)` ที่มี for วน times ครั้ง แล้วเรียก flash(150,3) · flash(450,3) · flash(150,3) · delay(2000)
Lab 2 · ทำนาย	9 — พอ +1 ได้ 10 เงื่อนไขเป็นจริงทันทีและรีเซ็ตเป็น 0 ก่อนถึงบรรทัดที่พิมพ์ จึงไม่เคยเห็น 10
Lab 2 · ⭐	`count = count + 5; if (count >= 100) count = 0;` — เห็น 5, 10, …, 95, 0
Lab 2 · ⭐⭐	เริ่ม `count = 10` · ลดทีละ 1 · ถ้า `count < 0` ให้ print("GO!") แล้วตั้งกลับเป็น 10
Lab 2 · ⭐⭐⭐	`int m = count / 60, s = count % 60; if (m < 10) oled.print('0'); oled.print(m); oled.print(':'); if (s < 10) oled.print('0'); oled.print(s);` และลบเงื่อนไขวนกลับที่ 10 ออก
Lab 3 · ทำนาย	**ความชื้น** — ไอน้ำจากลมหายใจเกาะเซนเซอร์ทันที ส่วนอุณหภูมิต้องรอให้ตัวเซนเซอร์ร้อนขึ้นก่อน
Lab 3 · ⭐	1000 เห็นการเปลี่ยนแปลงทันใจกว่า เหมาะกับการทดลอง · 5000 เหมาะกับการเก็บข้อมูลระยะยาว (ค่าอากาศไม่ได้เปลี่ยนเร็วขนาดนั้น)
Lab 3 · ⭐⭐	`Serial.print(t, 1); Serial.print(','); Serial.println(h, 1);`
Lab 3 · ⭐⭐⭐	จุดน้ำค้างโดยประมาณ (Magnus): γ = ln(h/100) + 17.62·t/(243.12+t) · Td = 243.12·γ/(17.62-γ) — ใช้ `log()` ใน Arduino
Lab 4 · ทำนาย	**แสดงต่อ** — โปรแกรมอยู่ในแฟลชของบอร์ด คอมแค่เป็นทางส่งโปรแกรมและจ่ายไฟ (ต่างจาก Serial Monitor ที่ต้องมีคอม)
Lab 4 · ⭐	`showOnOled("Humid %", h, 1);`
Lab 4 · ⭐⭐	ไม่ใช้ showOnOled — เขียนเอง: `clearDisplay` · setTextSize(2) · setCursor(0,0) print t · setCursor(0,32) print h · `display()`
Lab 4 · ⭐⭐⭐	0x44 = 68 · 0x3C = 60 · ใส่ 0x3D จอจะว่าง เพราะไม่มีอุปกรณ์ที่ที่อยู่นั้น — คำสั่งถูกส่งไปผิดบ้าน
Lab 5 · ทำนาย	25 × 1.8 = 45 → + 32 = **77 °F**
Lab 5 · ⭐	`float k = c + 273.15; showOnOled("Temp K", k, 1);`
Lab 5 · ⭐⭐	round(79.8) = 80 (ปัดตามหลักคณิต) · (int)79.8 = 79 (ตัดเศษทิ้ง)
Lab 5 · ⭐⭐⭐	(C+32)×1.8 - (1.8C+32) = 57.6 - 32 = **25.6 °F ทุกอุณหภูมิ** — ผิดคงที่เพราะเทอม C หักล้างกันหมด
Lab 6 · ทำนาย	ห้องปกติราว 25–28 °C ต่ำกว่า 30 → **ดับ**
Lab 6 · ⭐	`bool tooCold = (t < 24.0);`
Lab 6 · ⭐⭐	ใช้ h แทน t: `bool tooHumid = (h > 80.0);`
Lab 6 · ⭐⭐⭐	ประกาศ `bool alarm = false;` นอก loop · `if (t > 30.5) alarm = true; if (t < 29.5) alarm = false;` ช่วงกลางไม่แตะ alarm จึงคงสถานะเดิม (เรียกว่า hysteresis — แอร์และตู้เย็นใช้หลักนี้)
Lab 7 · ทำนาย	ห้องทั่วไป 25–28 °C / 55–70 %RH → ส่วนใหญ่ **ไม่มีเลย** · ห้องชื้นหลังฝนอาจติดช่อง 2

Lab 7 · ⭐	โดยทั่วไป 60–70 %RH ขึ้นไปถือว่าอับชื้น เชื้อราโตง่าย
Lab 7 · ⭐⭐	เก็บ `bool lastFan = false;` ไว้นอก loop · `if (fanOn != lastFan) { Serial.println(…); lastFan = fanOn; }`
Lab 7 · ⭐⭐⭐	`digitalWrite(RELAY_FAN, (fanOn && dryerOn) ? HIGH : LOW);`
Lab 8 · ทำนาย	ใต้โต๊ะส่วนใหญ่ได้ 30–150 lux ขึ้นกับตำแหน่งหลอดไฟ — คำตอบขึ้นกับห้อง ให้ดูที่การวัดจริง
Lab 8 · ⭐	ตั้งให้อยู่ระหว่างค่า “มืด” กับ “สว่าง” ที่จดไว้
Lab 8 · ⭐⭐	`int pct = constrain(map((long)lux, 0, 1000, 0, 100), 0, 100);`
Lab 8 · ⭐⭐⭐	เหมือนเฉลย ⭐⭐⭐ Lab 6: `if (lux < 80) on = true; if (lux > 120) on = false;` โดย on อยู่นอก loop
Lab 9 · ทำนาย	น้ำเปล่ามักได้สูงกว่า (ใกล้ 100 %) เพราะเป็นค่าที่ใช้สอบเทียบ “เปียก” · ดินที่เพิ่งรดน้ำมักเกิน 40 % ทั้งคู่ — ขึ้นกับดิน
Lab 9 · ⭐	กระบองเพชร 15–20 % · ผักใบ 50–60 %
Lab 9 · ⭐⭐	`long sum = 0; for (int i = 0; i < 10; i++) { sum += analogRead(SOIL_PIN); delay(5); } int raw = sum / 10;`
Lab 9 · ⭐⭐⭐	`if (dry) { digitalWrite(RELAY_PUMP, HIGH); delay(3000); digitalWrite(RELAY_PUMP, LOW); delay(30000); }` — ปั๊มจะไม่ค้างเปิดแม้เซนเซอร์หลุดจากดิน
Lab 10 · ทำนาย	ประมาณ **±1 g** (sin 90° = 1) — แรงโน้มถ่วงทั้งหมดย้ายมาแกน X
Lab 10 · ⭐	sin 45° ≈ **0.71**
Lab 10 · ⭐⭐	`bool level = (fabs(x) < 0.05);` แล้วให้ไฟติดเมื่อ level เป็นจริง
Lab 10 · ⭐⭐⭐	ขอ 4 ไบต์จาก 0x3B: X = ไบต์ 0–1 · Y = ไบต์ 2–3 · `tilted = fabs(x) > 0.5 || fabs(y) > 0.5` · มุม = `asin(constrain(x,-1,1)) * 180 / PI`
Lab 11 · ทำนาย	ห้องทั่วไปไม่ถึง 30 °C → **ไม่เปิด** แม้ชื้นเกิน 60 · ลด HOT_LIMIT ให้ต่ำกว่าอุณหภูมิห้อง
Lab 11 · ⭐	`digitalWrite(D1, (hot || humid) ? HIGH : LOW);` · ตาราง “หรือ”: F,F→F · F,T→T · T,F→T · T,T→T
Lab 11 · ⭐⭐	เพิ่ม `unsigned long lastBlink = 0; bool ledOn = false;` และที่ **ต้น** loop (ก่อนบรรทัด return): `if (millis() - lastBlink >= 500) { lastBlink = millis(); ledOn = !ledOn; digitalWrite(LED_BUILTIN, ledOn ? LOW : HIGH); }`
Lab 11 · ⭐⭐⭐	ประเมินจาก: ใช้ฟังก์ชันซ้ำได้ถูก · ขาไม่ชนกัน (D0/D1 เป็นรีเลย์ · D3 ดิน · D4/D5 I2C) · อธิบายเงื่อนไขได้

ภาคผนวก ง — สรุปการตรวจสอบเอกสารชุดนี้

- **คอมไพล์ผ่านทุกไฟล์** ด้วย arduino-cli · บอร์ด `esp32:esp32:XIAO_ESP32S3` · แพ็กเกจ esp32 3.3.10 · Adafruit SH110X 2.1.14 · เปิดคำเตือนทั้งหมด (`--warnings all`) ไม่มีคำเตือนจากโค้ดในใบงาน — สคริปต์สร้างเล่มคอมไพล์ใหม่ทุกครั้งก่อนสร้าง PDF และหยุดทันทีถ้าไฟล์ใดไม่ผ่าน โค้ดที่พิมพ์ในเล่มจึงเป็นไฟล์เดียวกับที่แจก
- **ขาและค่าต่าง ๆ ตรวจกับเฟิร์มแวร์ MakerFlow** ที่ทดสอบกับคิทจริงแล้ว (29 ก.ย. 2569): LED = GPIO21 active-low · รีเลย์ 1/2 = D0/D1 (HIGH = ทำงาน) · I2C = D4/D5 · คำสั่งวัด SHT30 (0x24 0x00) และสูตรแปลงค่า · BH1750 โหมด 0x10 หาร 1.2 · MPU ปลุกที่ 0x6B อ่านที่ 0x3B สเกล 16384/g · สูตรดินแห้ง/เปียก
- **ยังไม่ได้อัปโหลดสเก็ตช์ Arduino ลงบอร์ดจริง** (ไม่มีบอร์ดต่ออยู่ตอนจัดทำ) — ค่าตัวเลขใน “ผลที่ควรเห็น” มาจากการทดสอบฉบับ MakerFlow ด้วยเซนเซอร์ตัวเดียวกัน ครูควรอัปโหลด Lab 1 · 3 · 4 · 7 · 10 กับบอร์ด 1 ชุดก่อนสอน (ราว 10 นาที) โดยเฉพาะขั้นตอนเข้าโหมดดาวน์โหลดบนเครื่องของโรงเรียน
- Lab 9 (ความชื้นในดิน) ยังไม่ได้ลองกับเซนเซอร์ดินจริงทั้งสองฉบับ
- จุดต่างจากฉบับ MakerFlow ที่ตั้งใจ: SHT30 ในฉบับ Arduino **ไม่ตรวจ CRC** เพื่อให้โค้ดสั้นพอสำหรับผู้เริ่มต้น (เฟิร์มแวร์ MakerFlow ตรวจและทิ้งค่าที่เสียหาย) · Lab 1–10 ใช้ `delay()` เพราะเข้าใจง่าย แล้วค่อยสอน `millis()` ใน Lab 11

ขนาดโปรแกรม (พื้นที่แฟลชที่ใช้)

ไฟล์	ใช้แฟลช
`I2C_Scan.ino`	9 %
`Lab01_Blink.ino`	8 %
`Lab02_Counter.ino`	9 %
`Lab03_SHT30.ino`	9 %
`Lab04_OLED.ino`	9 %
`Lab05_Fahrenheit.ino`	9 %
`Lab06_HeatAlarm.ino`	9 %
`Lab07_Relays.ino`	9 %
`Lab08_AutoLight.ino`	9 %
`Lab09a_SoilRaw.ino`	8 %
`Lab09b_SoilPump.ino`	10 %
`Lab10_Tilt.ino`	9 %
`Lab11_Station.ino`	9 %