

คู่มือข้อมูลระบบและคู่มือใช้งาน

GNSS Monitoring Dashboard

ระบบเฝ้าระวังคุณภาพตำแหน่ง · อุณหภูมิ · ความสมบูรณ์ของข้อมูล · การแจ้งเตือนอัตโนมัติ

API v1 · Schema เวอร์ชันตามระบบรันจริง

จัดทำจากเนื้อหาหน้ากลาง “ข้อมูลระบบและคู่มือใช้งาน” ของ Dashboard

สารบัญ

- คุณค่าที่ Dashboard แสดง + หน่วยการวัด
- ความถี่การวัดและส่งข้อมูล
- รูปร่างกราฟแต่ละรูปแบบ (อ่านอย่างไร)
- การเรียก API — GET (อ่านข้อมูล) และ POST (ส่งข้อมูล + แจ้งเตือน)
- ทดสอบ GET/POST ด้วยมือ — ใช้ API Tester
- การจำลองการแจ้งเตือน (alert) ต่าง ๆ
- การตั้งค่าแจ้งเตือน LINE & Telegram — วิธีใช้
- ระบบสั่ง Run การตรวจสอบสถานะและแจ้งเตือนอัตโนมัติ (cron-job.org + Task Scheduler + Web autopilot)
- ตัวหนังสือและตัวอักษร — การออกแบบให้อ่านง่าย

ข้อมูลระบบและคู่มือใช้งาน

หน้านี้เป็นคู่มือกลางอัปเดตล่าสุดของทั้งระบบ — รวมทุกอย่างที่แก้ไขบน Dashboard: **ไม่ต้องใช้คีย์ Admin (X-Admin-Key) อีกต่อไป**, ตั้งค่า LINE/Telegram และ URL cron-job.org ได้จากหน้าเว็บ (DB override), ระบบสั่ง Run 3 ชั้นทำงาน 24 ชม. — พร้อมวิธีอ่านค่าต่าง ๆ, เรียก API แบบ GET/POST และส่ง/ทดสอบการแจ้งเตือนจริง

ระบบหลักทั้งหมด (ESP32 + LINE/Telegram + cron-job.org + web autopilot + Task Scheduler) ทำงานอยู่แล้วโดยไม่ต้องแก้ไขโค้ด — หน้านี้เป็นเพียงคู่มือและสารบบกลางของระบบ

คุณค่าที่ Dashboard แสดง + หน่วยการวัด

ตัวชี้วัด	หน่วย	จาก
อุณหภูมิภายใน (Internal)	°C (avg / min / max ต่อรอบ)	DS18B20 ตัวที่ 1
อุณหภูมิภายนอก (External)	°C (avg / min / max ต่อรอบ)	DS18B20 ตัวที่ 2
Delta อุณหภูมิ (int - ext)	°C	คำนวณจากทั้งคู่
สถานะตำแหน่ง (GPS fix)	Fixed / No fix	GNSS module
ดาวเทียมที่รับได้ / ใช้จำนวน	ดวง (visible / used)	GNSS module
ความแรงสัญญาณ C/N ₀	dB-Hz	GNSS module ต่อดาวเทียม
HDOP / VDOP / PDOP	ไม่มีหน่วย (เป็นอัตราส่วน)	GNSS module
ละติจูด / ลองจิจูด	องศาศกนิยม (°)	GNSS module
ระดับความสูง	เมตร (m)	GNSS module
มุม elevation / azimuth ดาวเทียม	° / ° (0–360)	GNSS module
ความแรง Wi-Fi RSSI	dBm	ESP32 Wi-Fi
หน่วยความจำ heap อิสระ	kB	ESP32
Uptime ต่อเนื่อง	นาที	ESP32
ความครบถ้วนข้อมูล (completeness)	% / จำนวนรอบ	คำนวณจากตารางรอบ 5 นาที

ค่าที่อ่านไม่ได้จะแสดงเป็น “-” (ไม่มีข้อมูล) ไม่ใช่ 0 เพื่อไม่ให้เข้าใจผิดว่าอุณหภูมิ 0 °C หรือพิกัด 0,0

ความถี่การวัดและส่งข้อมูล

ข้อมูล	อัตรา
GNSS (ตำแหน่ง/ดาวเทียม) อ่านล่วงหน้าสะสม	ประมาณ 1 ครั้งต่อวินาที
อุณหภูมิภายใน (DS18B20 #1)	ทุก 10 วินาที
อุณหภูมิภายนอก (DS18B20 #2)	ทุก 10 วินาที
สรุปและยิงขึ้น server (measurement)	ทุก 300 วินาที (5 นาที)
Dashboard รีเฟรชหน้าจอ	ทุก 30 วินาที (ค่าเริ่มต้น; เลือก 15s/30s/60s/5m)
cron-job.org / Windows Task Scheduler ตรวจสอบสถานะ + แจ้งเตือน	ทุก 5 นาที — ส่งจริงเฉพาะเมื่อสถานะเปลี่ยน (กันซ้ำ cooldown 900s)
Web autopilot (หน้า Dashboard ที่เปิดอยู่)	ทุกครั้งที่หน้ารีเฟรช (ใช้ marker <code>__AUTOPILOT__</code> กันซ้ำ)

Dashboard ไม่ถ่ายทอดค่าทุกวินาที — แสดงผลตามรอบสรุป 5 นาทีที่อุปกรณ์ส่งและรอบรีเฟรชของหน้าเว็บ ลองเปลี่ยนช่วงรีเฟรชได้ที่ปุ่มมุมซ้ายล่างหรือหน้า ตั้งค่าระบบ

รูปร่างกราฟแต่ละรูปแบบ (อ่านอย่างไร)

กราฟ	ลักษณะ / ข้อมูลที่วาด	อยู่หน้า
Skypilot	กราฟขั้ว (polar) ตำแหน่งดาวเทียมบนท้องฟ้า – มุม elevation ใช้รัศมี (กลาง = เหนือศีรษะ), มุม azimuth ใช้ตำแหน่งรอบวง, ขนาดจุด/สี = ความแรงสัญญาณ, มี 2D + 3D	ภาพรวม
Globe 3D	ทรงกลมหมุน 3 มิติ ตำแหน่งดาวเทียมและเส้นขอบฟ้า	ภาพรวม
จำนวนข้อมูลรายชั่วโมง	กราฟแท่ง (bar) นับจำนวนรอบต่อชั่วโมง	กราฟวิเคราะห์
อุณหภูมิรายวัน	เส้น (line) min-avg-max °C รายวัน (3 เส้น = 3 ค่าคงที่)	กราฟวิเคราะห์ / ค่า Data สถิติรายวัน
อุณหภูมิภายใน-ภายนอก	เส้น 2 เส้น – ภายใน • ภายนอก ประ, แกน Y = °C, เวลาตาม X	กราฟวิเคราะห์ / ค่า Data เติม
Delta อุณหภูมิ	เส้นค่าต่าง (internal - external) °C	กราฟวิเคราะห์
ความแรงสัญญาณรายกลุ่มดาว	เส้น/จุด (line/scatter) C/No (dB-Hz) แยกด้วยสีต่อกลุ่มดาว GPS/GLONASS/GALILEO/BEIDOU	กราฟวิเคราะห์
heap / WiFi	เส้น – heap (kB) และ RSSI (dBm) + uptime (นาท)	กราฟวิเคราะห์
เปรียบเทียบ 2 วัน	เส้นซ้อน 2 ชุด (วันที่ ก vs ข) สำหรับอุณหภูมิ ความครบถ้วน กลุ่มดาว	ค่า Data เปรียบเทียบ 2 วัน
ความครบถ้วน (completeness)	แท่ง/เส้น % ข้อมูลครบตามรอบ 5 นาที + ช่องว่าง (gap) เฉพาะช่วง	กราฟวิเคราะห์ / เปรียบเทียบ

ปุ่ม  (Graph Settings) มุมซ้ายล่างปรับเส้นหนา/พื้นที่ใต้เส้น/แสดงจุด/สีของทุกกราฟได้ทันที โดยไม่ต้องแก้ไขโค้ด

รูปแบบการเรียก API – GET (อ่านข้อมูล)

Base URL: `https://project12569.theeiot.com/EE67172310228/api/v1`

endpoint (GET)	พารามิเตอร์ที่ใช้
<code>latest.php</code>	– (ค่าล่าสุดของ device)
<code>measurements.php</code>	<code>limit</code> , <code>page</code> , <code>start_time</code> , <code>end_time</code>
<code>temperature.php</code>	<code>start_time</code> , <code>end_time</code>
<code>satellites.php</code>	<code>start_time</code> , <code>end_time</code> , <code>constellation</code> , <code>prn</code> , <code>points</code>
<code>skypilot.php</code>	<code>timestamp</code> , <code>constellation</code>
<code>satellite-track.php</code>	<code>constellation</code> , <code>prn</code> , <code>start_time</code> , <code>end_time</code>
<code>statistics.php</code>	<code>start_time</code> , <code>end_time</code>
<code>constellations.php</code>	<code>start_time</code> , <code>end_time</code>

endpoint (GET)	พารามิเตอร์ที่ใช้
system-health.php	start_time, end_time
data-completeness.php	start_time, end_time, interval (≥30s)
devices.php	include
api-health.php	— (สถานะระบบ)
api-logs.php	method, http_status, endpoint, limit, page
notifications.php	— (การตั้งค่าแจ้งเตือน + ที่มา credential แบบ mask)
scheduler-status.php	— (สถานะระบบสั่ง Run: last run / healthy / รอบล่าสุด / พาส่ง / URL ที่ใช้)
scheduler-config.php	— (URL cron ที่ใช้จริง + ต้นทาง db/default)
alert-check.php	device — ตรวจจับสถานะ + ส่งแจ้งเตือนทันที (ไม่มี key)
export-csv.php / export-excel.php	type, start_time, end_time, constellation, prn

ทุก GET ใช้ ?device=GNSS01 ถ่ายภาพ URL เสมอ และพารามิเตอร์เวลาใช้ **ISO8601 UTC** เช่น 2026-09-11T05:15:00Z
 — endpoints ส่วนแจ้งเตือน/สั่ง Run (notifications / scheduler-status / scheduler-config / alert-check)
 ไม่ต้องใช้ key

รูปแบบการเรียก API — POST (ส่งข้อมูลจากอุปกรณ์ + แจ้งเตือน)

endpoint	วิธีใช้
gnss.php	อัปโหลด measurement จาก ESP32 — ต้องมี header Content-Type: application/json + X-API-Key: <device key>
notifications.php	บันทึกการเปิด/ปิดช่องทาง + อัปโหลด/ล้าง credential — alerts_enabled, line_enabled, telegram_enabled + 4 ช่องค่า (ส่ง "" = ล้าง)
notify-test.php?channel=line telegram all	ส่งข้อความทดสอบจริงไปยังช่องทาง (ไม่ต้อง key)
status-send.php?device=GNSS01	ส่งสถานะปัจจุบันไป LINE + Telegram ทันที (ไม่ต้อง key)

ตัวอย่าง body ของ gnss.php

```
{
  "uuid": "a1b2c3d4-...-9f10",
  "device": "GNSS01",
  "timestamp": "2026-09-15T04:00:00Z",
  "fix": true,
  "lat": 13.750000, "lon": 100.500000, "alt": 5.2,
  "hdop": 0.8, "vdop": 1.1, "pdop": 1.4,
  "satellites_visible": 12, "satellites_used": 9,
  "satellite_summary": [{ "constellation": "GPS", "prn": 3, "signal_db_hz": 42.5 }],
  "temperature": { "internal_c": 34.2, "external_c": 31.8 },
  "system": { "wifi_rssi_dbm": -52, "free_heap_bytes": 180224,
    "minimum_free_heap_bytes": 150000, "uptime_s": 5400,
    "reboot_count": 1, "wifi_disconnect_count": 0,
```

```
"http_fail_count": 0 },
"fw_version": "v1.0.3"
}
```

Response ของ `gnss.php` : 201 สร้างใหม่ · 200 (duplicate uuid) · 400/401/405/415/422/429 · ตัวกระตุ้นแจ้งเตือนอื่น (notifications / notify-test / status-send / alert-check / alert-cron / scheduler-config) **ไม่ต้องใช้ key** — Rate limit: POST 60 ครั้ง/นาท/device · GET 120 ครั้ง/นาท/IP (ใน `config/security.php`)

ทดสอบ GET/POST ด้วยมือ — ใช้ API Tester

เปิดหน้า “**API Tester Arduino IDE**” (?sec=tester) แล้วทำตามนี้:

1. Device: `GNSS01`
2. **API Key**: กรอก **X-API-Key ของ device** (ดูการ์ด “ข้อมูล Credentials (API — Arduino/ESP32)”) — ใช้เฉพาะ POST `gnss.php` ; endpoint อื่นไม่ต้อง key
3. Tab **GET API**: เลือก endpoint → กรอกพารามิเตอร์ → กด **Send GET request**
4. Tab **POST API**: วาง JSON ตัวอย่าง → กด **Send POST Measurement**
5. อ่านผลในช่อง Response + Request history (ทุกครั้งอัตโนมัติ)

Key ของระบบนี้:

X-API-Key (device) `e312439187c981d517487bb6cd40ed3b0eeb6d7e18155dce48a10d666b36d4ff`

device key = key ตัวเดียวกับที่ ESP32 ใช้จาก `config/device_api_key.txt` (สำหรับทดสอบ POST จำลอง) — หน้านี้เปิดออนไลน์ **ห้ามแชร์ URL นี้กับใคร** · ไม่มีคีย์ `X-Admin-Key` อีกต่อไป

การจำลองการแจ้งเตือน (alert) ต่าง ๆ

ใช้หน้า “Setting and Tester การแจ้งเตือนอัตโนมัติ LINE & Telegram” (?sec=notify) — ไม่ต้องแก้โค้ดและไม่ต้องใช้ key:

- **ทดสอบช่องทาง**: ปุ่ม **ทดสอบ LINE / ทดสอบ Telegram / ทดสอบช่องทางที่เปิด** ส่งข้อความจริงไปยัง LINE/Telegram
- **แจ้งเตือนสถานะทันที**: ปุ่ม **แจ้งเตือนสถานะปัจจุบัน** ส่งสถานะ GNU (ONLINE/WARNING/OFFLINE) ในขณะนั้น
- **จำลองผ่าน web autopilot**: ทุกครั้งที่หน้ารีเฟรช ระบบจะรัน autopilot — สถานะเปลี่ยน OFFLINE / WARNING / ONLINE / NO FIX / temp error จะส่ง LINE+Telegram อัตโนมัติ (กันซ้ำ cooldown 900s)
- **เช็คประวัติจริง**: ส่วน “การตรวจสอบด้วยตัวเอง” (?sec=scheduler) มีคำสั่ง SQL ดูตาราง `alert_events` (status= `SENT` / `AUTOPILOT`)

ระบบแจ้งเตือน LINE/Telegram + cron-job.org + ตัวตั้งเวลา เป็นของเดิมที่ทำงานอยู่ — การตั้งค่าทั้งหมดย้ายมาอยู่บน Dashboard แล้ว (ดูหัวข้อถัดไป)

การตั้งค่าแจ้งเตือน (LINE & Telegram) — วิธีใช้ (หัวข้อ “การตั้งค่าแจ้งเตือน”)

เดิมเคยต้องพิมพ์คีย์ Admin เพื่อส่งระบบ — ตอนนี้จัดการทุกอย่างจากหน้าเว็บ **โดยไม่มีคีย์ใด ๆ**:

รายการ	วิธีใช้
เปิด/ปิดช่องทาง	ติ๊ก LINE Notification / Telegram Notification แล้วกด บันทึกการเปิดใช้งานการแจ้งเตือน — บันทึกจริงลง DB (<code>notification_settings</code>) ใช้แจ้งเตือนทันที
อัปโหลด credential	กรอกครบ 4 ช่อง (<code>LINE_CHANNEL_ACCESS_TOKEN</code> · <code>LINE_TARGET_ID</code> · <code>TELEGRAM_BOT_TOKEN</code> · <code>TELEGRAM_CHAT_ID</code>) → กด อัปโหลด credentials + บันทึกการตั้งค่า — เก็บลงตาราง <code>notification_credentials</code> (DB override)
ช่องที่เว้นว่าง	เก็บค่าที่มีอยู่เดิมไว้ (ไม่อัปโหลด/ไม่ล้าง)
ล้างกลับค่า config	กด ล้างค่า LINE / ล้างค่า Telegram — ช่องนั้นกลับไปใช้ค่าเริ่มต้นใน <code>config/notifications.php</code> (ส่ง <code>""</code>)
ทดสอบ / ส่งสถานะ	ปุ่ม ทดสอบ LINE / ทดสอบ Telegram / ทดสอบช่องทางที่เปิด / แจ้งเตือนสถานะปัจจุบัน
สถานะที่แสดง	● เปิดใช้งาน · ● ปิดใช้งาน — แยก Enabled (เปิดใช้) กับ Configured (มี credential) · ค่าเป็น secret แสดงแบบ mask (หัว...ท้าย) เท่านั้น

ลำดับการใช้ค่า: ค่าที่อัปโหลดจาก Dashboard (DB) ใช้ก่อนเสมอ — ถ้าค่าใน DB ว่าง → ระบบใช้ค่าใน `config/notifications.php` โดยอัตโนมัติ

ระบบสั่ง Run การตรวจสอบสถานะระบบและแจ้งเตือนอัตโนมัติ — การตั้งค่า & สถานะ (?sec=scheduler)

หน้า “ระบบสั่ง Run” (cron-job.org + Windows Task Scheduler + CLI) — ตั้งค่า URL ที่ cloud เรียกได้จากหน้าเว็บ (DB override) และดูสถานะการทำงานจริงได้ทันที:

URL เริ่มต้นอัตโนมัติ (default) `<scheme>://<host> + app_base() + /api/v1/alert-cron.php`

URL ที่ใช้จริงตอนนี้ `https://project12569.theeiot.com/EE6717...ert-cron.php`

(ใช้ค่าเริ่มต้นอัตโนมัติ) (จาก `scheduler-config.php`)

- แก้ URL:** กรอก URL ใหม่ (เช่น จากหน้า cron-job.org — ใส่ key ต่อท้ายได้) ในช่อง “แก้ URL ที่ใช้จริง” → กด **บันทึกค่าใหม่** (เก็บในตาราง `scheduler_settings`)
- คืนค่าเริ่มต้น:** กด **เคลียร์ (ใช้ค่าเริ่มต้น)** — ค่าที่กรอกถูกลบ ระบบกลับไปใช้ค่า default อัตโนมัติ
- เช็คสถานะ:** กด **รีเฟรชสถานะ** — ดู LAST RUN / AGE / HEALTHY (วงจรรอบ $\leq 660s$), ตาราง 8 รอบล่าสุด ★ cron-job.org และผลที่ส่ง LINE/TG

ชั้น	ตัวรับ	วิธีทำงาน	ส่งซ้ำ?
A	Web autopilot (หน้าเว็บที่เปิดอยู่)	ทุกครั้งที่หน้านี้โหลดข้อมูล จะเรียก <code>alert-check.php</code> อัตโนมัติ (ไม่ต้องใช้ key)	ไม่ — กันด้วย <code>__AUTOPILOT__</code> + cooldown
B	Windows Task Scheduler + CLI	เครื่อง PC ที่เปิดตลอด ใช้ <code>php alert-check-cli.php</code> ทุก 5 นาที (Task <code>GNSS-AlertCheck</code>)	ไม่ — ใช้ marker ตัวเดียวกับชั้น A
C	cron-job.org (cloud ฟรี)	เรียก <code>api/v1/alert-cron.php</code> ตาม URL ที่ตั้งไว้ด้านบน ทุก 5 นาที (ไม่ต้องใช้ key)	ไม่ — ใช้ engine ตัวเดียวกัน + GET_LOCK 60s

การส่งจริงทั้งหมดอยู่ใน `includes/alert-engine.php` เดียวกัน — ส่งเฉพาะเมื่อสถานะเปลี่ยน (ONLINE/WARNING/OFFLINE/NO FIX/temp error) ภายใน cooldown 900 วินาที จึงไม่มีทางแจ้งซ้ำแม้หลายตัว กระตุ้น · คู่มือครบทั้งระบบอยู่ใน PDF “ขั้นตอนการตั้งค่าทั้งหมด (ละเอียด)” ที่มาต่อจากหัวข้อ “การตั้งค่า” ด้านบน

ตัวหนังสือและตัวอักษร — อ่านง่าย

- ฟอนต์ **Prompt** (Google Fonts) ทั้งหมด + ฟอนต์ mono สำหรับค่าตัวเลข/code เพื่อแยกแยะข้อมูลชัดเจน
- การ์ด KPI ใหญ่ อ่านค่าจากระยะไกลได้; ตารางมีหัวตารางชัดเจน; “-” แทนข้อมูลขาด (ไม่ปนกับค่า 0)
- ธีมพื้นหลังสีเข้มตัดกับสีสัญญาณ (เขียว = ปกติ / เหลือง = เตือน / แดง = ผิดปกติ)
- ปรับเพิ่ม/ลดขนาดกราฟได้ด้วยปุ่ม ☺ และหน้าปรับงานตามขนาดจอ (responsive)

เอกสารแบบเพิ่มเติม (PDF ที่ฝังไว้ใน Dashboard)

คู่มือละเอียดอีก 2 ฉบับที่หน้า Dashboard เปิดในกรอบ iframe ภายในหัวข้อการตั้งค่า มีไฟล์แยก:

- `GNSS_LINE_Telegram_Setting_Tester_Guide_REVISIED.pdf` — คู่มือตั้งค่าระบบแจ้งเตือน LINE & Telegram (ทดสอบจริง + ตัวอย่าง credential)
- `GNSS_AutoAlert_RUN_CronJob_REVISIED.pdf` — คู่มือขั้นตอนการจัดทำระบบ GNSS + ESP32 + CronJob + เพิ่มเซ็นเซอร์อุณหภูมิ

รวมกับเนื้อหาหน้าหน้านี้ (คู่มือกลางอัปเดตล่าสุด) จึงครอบคลุมทั้งการอ่านค่า, เรียก API, ตั้งค่าแจ้งเตือน และระบบสั่ง Run อัตโนมัติทั้งหมด