

מערכת רישום מתח – ESP32 תיעוד מלא ⚡

סקירה כללית 📋

מערכת מתקדמת לרישום נתוני מתח בעזרת ESP32, הכוללת:

- שליחת נתונים לשרת FastAPI
- שמירתם במסד PostgreSQL
- ניתוח וחישוב ממוצעים יומיים
- מעקב אחר טריגרים מהשטח

קישורים חשובים 🌐

כתובות שרת: 🔗

- שרת ראשי 🟢 : <https://esp32-voltage-logger.onrender.com>
- Swagger: /docs 📘
- ReDoc: /redoc 📖

מסד נתונים: (PostgreSQL) 🗄️

- URL:
postgresql://postgres_voltage_logger:QhQvIgtV2svJj5CwmBKxDthLfS5GzUxY@dpg-
d1n4p8ndiees73eoc7h0-a.oregon-postgres.render.com/voltage_logger
- שירות: Render
- אזור: Oregon, USA

esp32/

└─ include/

| └─ config.h, wifi_manager.h, ...

└─ src/

| └─ main.cpp, http_client.cpp, ...

└─ platformio.ini

הגדרות פינים:

TRIGGER_PIN_START = 15

TRIGGER_PIN_EVENT = 4

VOLTAGE_PIN = 34

LED_PIN = 2

WIFI_RESET_PIN = 16

הגדרות זמנים:

SAMPLING_DELAY_MS = 1000

SAMPLING_START_DELAY_MS = 24000

TOTAL_TIMER_DURATION_MS = 30000

MAX_SAMPLES = 6

תהליך עבודה:

1. אתחול רכיבים
 2. קבלת טריגר
 3. התחלת טיימר
 4. דגימה כל שנייה (אחרי 24 שניות)
 5. שליחת הנתון הגבוה ביותר לשרת
 6. שליחת אירועים מיידיים בטריגר נוסף
-

חלק 2: שרת FastAPI (Python)

מבנה תיקיות: 📁

```
backend/
├── src/
│   ├── main.py, config.py, ...
│   ├── db/, models/, schemas/, services/, routers/, tasks/
│   ├── requirements.txt
│   ├── Dockerfile
└── docker-compose.yml
```

מסד נתונים: 🗄️

voltage_logs (id, timestamp, max_voltage, ...)

trigger_events (id, event_time, ...)

daily_averages (id, date, average, ...)

API – 📡 נקודות קצה

שיטה תחום	נתיב	תיאור
POST מתח	/api/v1/voltage/	שליחת נתון
GET	/api/v1/voltage/	כל הנתונים
GET	/api/v1/voltage/{id}	לפי מזהה
GET	/api/v1/voltage/latest/	האחרון
POST אירועים	/api/v1/event/	שליחת אירוע
GET	/api/v1/event/	כל האירועים
GET	/api/v1/event/{id}	לפי מזהה
GET	/api/v1/event/times/	רשימת זמנים
GET ממוצעים	/api/v1/daily-average/	כל הממוצעים
GET	/api/v1/daily-average/latest/	ממוצע אחרון

תיאור	נתיב	שיטה תחום
חישוב ידני	/api/v1/daily-average/calculate/	POST
בדיקת חיבור	/	GET כללי
בדיקת בריאות	/health	GET
Swagger	/docs	GET
ReDoc	/redoc	GET

דוגמאות JSON

שליחת נתון מתח: 

```
{
  "timestamp": "2025-01-06T12:34:56",
  "max_voltage": 3.45
}
```

שליחת אירוע: 

```
{
  "event_time": "2025-01-06T12:34:56"
}
```

תגובה: 

```
{
  "status": "success",
  "message": "Voltage data saved successfully",
  "data": {
    "id": 1,
    "timestamp": "2025-01-06T12:34:56",
    "max_voltage": 3.45,
    "created_at": "2025-01-06T12:34:56"
  }
}
```

התקנה והגדרה

דרישות:

- Python 3.11+
- Docker & Docker Compose
- PlatformIO
- ESP32 לוח

התקנת Backend עם Docker:

```
git clone <repository-url>
cd esp32_voltage_logger
docker-compose up -d
```

התקנה מקומית:

```
cd backend
pip install -r requirements.txt
python src/main.py
```

צריבת ESP32 עם PlatformIO:

```
cd esp32
pio run --target upload
pio device monitor
```

הגדרת WiFi ל-ESP32

- 1. חבר את הבקר לחשמל
- 2. התחבר ל - ESP32_VoltageLogge
- 3. סיסמה: 12345678
- 4. דף הגדרה ייפתח אוטומטית

קובץ .env

```
DATABASE_URL=postgresql://...  
APP_ENVIRONMENT=production  
APP_DEBUG=false
```

זרימת נתונים

→ ESP32  שרת

- 1. טריגר
- 2. דגימה
- 3. חישוב
- 4. שליחה
- 5. שמירה במסד

עיבוד בצד השרת

- אימות → שמירה → תגובה ב־JSON

ממוצע יומי

- תזמון יומי: 00:01
- חישוב ממוצע
- שמירה בטבלה נפרדת

בדיקות ולוגים

בדיקות:

cd backend

pytest

לוגים:

docker-compose logs backend

pio device monitor

ביצועים ואבטחה

ביצועים: ⚡

- דגימה כל שנייה, שליחה כל 30 שניות
- שרת תומך במאות קריאות לדקה
- אינדוקס על שדות חשובים במסד

אבטחה:

- אימות נתונים עם Pydantic
 - בקרת גישה עם CORS
 - טיפול בשגיאות ולוגים
-

פתרון בעיות




ESP32:

- WiFi לא מתחבר → בדוק רשת וסיסמה
- אין דגימה → בדוק פין 36
- אין שליחה → ודא כתובת שרת תקינה

שרת:

- לא שומר למסד → בדוק הגדרת DATABASE_URL
- שגיאות → API בדוק לוגים
- בעיות → CORS בדוק middleware

תמיכה

-  מפתח: דניאל שמש
-  אימייל ds5121h@gmail.com
-  GitHub: github.com/Ds8200

רישיון

פרויקט זה מופץ תחת רישיון MIT.
לפרטים נוספים ראה קובץ LICENSE.

עודכן לאחרונה: 15/07/2025
