SmartCompactorLogger תיעוד מלא ומפורט - מערכת

תוכן עניינים

מבוא כללי

מטרת המערכת

מערכת **SmartCompactorLogger** היא פלטפורמה חכמה לניטור, רישום וניהול של פחי אשפה עם דחסן חשמלי. המערכת מורכבת משני חלקים עיקריים המתקשרים זה עם זה בזמן אמת (קומפקטור):

- **ניטור בזמן אמת** של פחי אשפה עם דחסן חשמלי
- **חיסכון במשאבים** על ידי זיהוי מדויק של זמני פינוי נדרשים
- **תחזוקה חכמה** עם זיהוי תקלות ועומסים
- **שקיפות מלאה** של כל הנתונים עם יכולת ניתוח מתקדמת

רכיבי המערכת

רכיב	תיאור	טכנולוגיה
בקר ESP32	מותקן פיזית בפח, מודד מתחים, מזהה אירועים	C++, Arduino Framework
שרת Backend	API מקבל נתונים, מעבד, שומר ומספק	Python, FastAPI, SQLAlchemy
מסד נתונים	אחסון כל הנתונים וההיסטוריה	PostgreSQL
ממשק ניהול	לניהול מכשירים ודוחות web דף	HTML, CSS, JavaScript

---

ארכיטקטורת המערכת

מבנה תיקיות

```
SmartCompactorLogger/
├── esp32_controler/
│   ├── src/
│   │   ├── main.cpp
│   │   ├── config.cpp
│   │   ├── wifi_manager.cpp
│   │   ├── compression_sensor.cpp
│   │   ├── trigger_manager.cpp
│   │   ├── trigger_handler.cpp
│   │   ├── event_sender.cpp
│   │   ├── led_manager.cpp
│   │   ├── time_utils.cpp
│   │   ├── location_sync.cpp
│   │   ├── config_manager.cpp
│   │   └── web_config_server.cpp
│   ├── include/
│   │   ├── config.h
│   │   ├── globals.h
│   │   ├── wifi_manager.h
│   │   ├── compression_sensor.h
│   │   ├── trigger_manager.h
│   │   ├── trigger_handler.h
│   │   ├── event_sender.h
│   │   ├── led_manager.h
│   │   ├── time_utils.h
│   │   ├── location_sync.h
│   │   ├── config_manager.h
│   │   └── web_config_server.h
│   └── platformio.ini
├── server/
│   ├── src/
│   │   ├── main.py
│   │   ├── config.py
│   │   ├── constants.py
│   │   ├── lifespan.py
│   │   ├── routers/
│   │   ├── models/
│   │   ├── schemas/
│   │   ├── services/
│   │   ├── utils/
│   │   ├── db/
│   │   ├── tasks/
│   │   └── templates/
│   ├── docker-compose.yml
```

```
│   └── init.sql
└── README.md
```

טכנולוגיות

קטגוריה	טכנולוגיה	שימוש
בקר	ESP32, Arduino Framework, C++	חיישנים, תקשורת
שרת	FastAPI, SQLAlchemy, PostgreSQL	API, עיבוד נתונים
תזמון	APScheduler	משימות רקע
ממשק	HTML, CSS, JavaScript	דפי ניהול
איחסון	Docker, PostgreSQL	בסיס נתונים

קישורים רלוונטיים

קבצי קוד עיקריים

- **קוד ESP32**: `esp32_controler/src/`
- **קוד שרת**: `server/src/`
- **דוקומנטציה API**: http://62.90.195.107/docs
- **קובץ Docker**: `server/docker-compose.yml`
- **קונפיגורציה ESP32**: `esp32_controler/platformio.ini`
- **קונפיגורציה שרת**: `server/src/config.py`
- **README ראשי**: `README.md`

עיקריים API נתיבי

נתיב	שיטה	תיאור
`/api/v1/devices/location`	POST	שליחת מיקום
`/api/v1/compressions`	POST	אירוע דחיסה
`/api/v1/emptying-events`	POST	אירוע פינוי
`/api/v1/devices`	GET	רשימת מכשירים
`/api/v1/daily-average`	GET	ממוצעים יומיים

תרשים זרימה כללי

זרימת מידע בסיסית

```
mermaid
```

```
graph TD
  A[אתחול בקר] --> B[טעינת קונפיגורציה]
  B --> C[חיבור WiFi]
  C --> D[סנכרון NTP]
  D --> E[שליחת מיקום ראשוני]
  E --> F[שרת מקבל]
  F --> G[שומר במסד]
  G --> H[מחזיר אישור]
  H --> I[לולאת עבודה]
  I --> J[בדיקת טריגרים]
  J --> K[דגימת מתח]
  K --> L[שליחת אירועים]
  L --> M[עיבוד בשרת]
  M --> N[קבלת נתונים]
  N --> O[חישוב קיבולת]
  O --> P[שמירה במסד]
  P --> Q[משימות רקע]
  Q --> R[חישוב ממוצעים יומיים]
  R --> S[עדכון דוחות]
```

תרחישי עבודה

תרחיש	תהליך
דחיסה	טריגר → דגימת מתח → חישוב מקסימום → שליחה לשרת
פינוי	טריגר → שליחת אירוע → איפוס מונה דחיסות
RESET	קונפיגורציה חדשה → AP לחצן פיזי → מעבר למצב
שגיאה	ניסיון חוזר → ריפוי אוטומטי → לוגים מפורטים

מסד נתונים

טבלאות עיקריות

1. devices

שדה	טיפוס	תיאור
id	VARCHAR(64)	מזהה ייחודי של המכשיר (Primary Key)
name	VARCHAR(128)	שם המכשיר (אופציונלי)
latitude	FLOAT	קו רוחב
longitude	FLOAT	קו אורך
location_address	VARCHAR(256)	כתובת מיקום
last_emptying_timestamp	TIMESTAMP	זמן פינוי אחרון
compression_count	INTEGER	מונה דחיסות מאז פינוי אחרון
created_at	TIMESTAMP	זמן יצירה
updated_at	TIMESTAMP	זמן עדכון אחרון

2. compression_events

שדה	טיפוס	תיאור
id	INTEGER	מזהה אירוע (Primary Key, Auto Increment)
device_id	VARCHAR(64)	מזהה מכשיר (Foreign Key)
timestamp	TIMESTAMP	זמן האירוע
voltage_max	FLOAT	מתח מקסימלי שנמדד
capacity_in_percent	FLOAT	אחוז קיבולת מחושב
created_at	TIMESTAMP	זמן יצירה
updated_at	TIMESTAMP	זמן עדכון

3. emptying_events

שדה	טיפוס	תיאור
id	INTEGER	מזהה אירוע (Primary Key, Auto Increment)
device_id	VARCHAR(64)	מזהה מכשיר (Foreign Key)
timestamp	TIMESTAMP	זמן הפינוי
created_at	TIMESTAMP	זמן יצירה
updated_at	TIMESTAMP	זמן עדכון

4. daily_averages

שדה	טיפוס	תיאור
`id`	INTEGER	מזהה (Primary Key, Auto Increment)
`device_id`	VARCHAR(64)	מזהה מכשיר (Foreign Key)
`date`	DATE	תאריך (יום)
`average`	FLOAT	ממוצע מתח יומי
`calculated_at`	TIMESTAMP	זמן חישוב
`created_at`	TIMESTAMP	זמן יצירה
`updated_at`	TIMESTAMP	זמן עדכון

אינדקסים

- `idx_voltage_samples_device_id` - על `device_id` בטבלת `compression_events`
- `idx_voltage_samples_timestamp` - על `timestamp` בטבלת `compression_events`
- `idx_daily_averages_device_id` - על `device_id` בטבלת `daily_averages`
- `idx_daily_averages_date` - על `date` בטבלת `daily_averages`
- `uix_device_date` - ייחודיות על `(device_id, date)` בטבלת `daily_averages`

API Endpoints

1. ניהול מכשירים

POST /api/v1/devices/location

תיאור: עדכון מיקום מכשיר

קלט:

```
json
{
  "device_id": "string",
  "latitude": 0.0,
  "longitude": 0.0
}
```

פלט:

```
json
```

```
{
  "id": "string",
  "name": "string",
  "latitude": 0.0,
  "longitude": 0.0,
  "location_address": "string",
  "last_emptying_timestamp": "2025-07-27T10:00:00Z",
  "compression_count": 0,
  "created_at": "2025-07-27T10:00:00Z",
  "updated_at": "2025-07-27T10:00:00Z"
}
```

GET /api/v1/devices

תיאור: קבלת רשימת מכשירים עם דגימות אחרונות

פלט:

```json
[
  {
    "device": {
      "id": "string",
      "name": "string",
      "latitude": 0.0,
      "longitude": 0.0,
      "location_address": "string",
      "last_emptying_timestamp": "2025-07-27T10:00:00Z",
      "compression_count": 0,
      "created_at": "2025-07-27T10:00:00Z",
      "updated_at": "2025-07-27T10:00:00Z"
    },
    "recent_compressions": [
      {
        "id": 0,
        "timestamp": "2025-07-27T10:00:00Z",
        "voltage_max": 0.0,
        "capacity_in_percent": 0.0
      }
    ]
  }
]
```

2. אירועי דחיסה

POST /api/v1/compressions

תיאור: שליחת אירוע דחיסה

קלט:

```
json
{
  "device_id": "string",
  "timestamp": "2025-07-27T10:00:00Z",
  "voltage_max": 0.0
}
```

פלט:

```
json
{
  "id": 0,
  "device_id": "string",
  "timestamp": "2025-07-27T10:00:00Z",
  "voltage_max": 0.0,
  "capacity_in_percent": 0.0,
  "created_at": "2025-07-27T10:00:00Z",
  "updated_at": "2025-07-27T10:00:00Z"
}
```

3. אירועי פינוי

POST /api/v1/emptying-events

תיאור: שליחת אירוע פינוי

קלט:

```
json
{
  "device_id": "string",
  "timestamp": "2025-07-27T10:00:00Z"
}
```

פלט:

```
json
```

```
{
  "id": 0,
  "device_id": "string",
  "timestamp": "2025-07-27T10:00:00Z",
  "created_at": "2025-07-27T10:00:00Z",
  "updated_at": "2025-07-27T10:00:00Z"
}
```

4. ממוצעים יומיים

GET /api/v1/daily-average

תיאור: קבלת ממוצעים יומיים עם פאגינציה **פרמטרים**: `skip=0&limit=100`

פלט:

```json
[
  {
    "id": 0,
    "device_id": "string",
    "date": "2025-07-27",
    "average": 0.0,
    "calculated_at": "2025-07-27T10:00:00Z",
    "created_at": "2025-07-27T10:00:00Z",
    "updated_at": "2025-07-27T10:00:00Z"
  }
]
```

GET /api/v1/daily-average/latest

תיאור: קבלת הממוצע היומי האחרון

POST /api/v1/daily-average/calculate

תיאור: חישוב ידני של ממוצע יומי **קלט**: `date=2025-07-21`

5. בריאות המערכת

GET /api/v1/health

תיאור: בדיקת בריאות המערכת

פלט:

```json
```

```
{
  "status": "healthy",
  "database": "connected",
  "scheduler": "running",
  "timestamp": "2025-07-27T10:00:00Z"
}
```

בקר ESP32 - קוד C++

מבנה קבצים

קבצי Header (include/)

קובץ	תיאור
config.h	הגדרות מערכת, פינים, קבועים, הודעות
globals.h	אובייקטים גלובליים
wifi_manager.h	ניהול חיבור WiFi
compression_sensor.h	דגימת מתח וחישובים
trigger_manager.h	ניהול טריגרים
trigger_handler.h	טיפול בטריגרים
event_sender.h	שליחת אירועים לשרת
led_manager.h	ניהול סטטוס LED
time_utils.h	ניהול זמן ו-NTP
location_sync.h	סנכרון מיקום
config_manager.h	ניהול קונפיגורציה
web_config_server.h	שרת קונפיגורציה web

קבצי מקור (src/)

קובץ	תיאור
main.cpp	קובץ ראשי, אתחול ולולאה
config.cpp	מימוש הגדרות ברירת מחדל
wifi_manager.cpp	מימוש ניהול WiFi
compression_sensor.cpp	מימוש דגימת מתח
trigger_manager.cpp	מימוש ניהול טריגרים
trigger_handler.cpp	מימוש טיפול בטריגרים
event_sender.cpp	מימוש שליחת אירועים
led_manager.cpp	מימוש ניהול LED
time_utils.cpp	מימוש ניהול זמן
location_sync.cpp	מימוש סנכרון מיקום
config_manager.cpp	מימוש ניהול קונפיגורציה
web_config_server.cpp	מימוש שרת קונפיגורציה

הגדרות מערכת (config.h)

פינים (Pins)

פין	מטרה	ערך
TRIGGER_PIN_START	פין דיגיטלי לטריגר התחלה	15
TRIGGER_PIN_EVENT	פין דיגיטלי לטריגר אירוע	4
VOLTAGE_PIN	פין אנלוגי לקריאת מתח	34
LED_PIN	מובנה לסטטוס LED	2
WIFI_RESET_PIN	פין לאיפוס WiFi	16

זמנים (Timing)

קבוע	תיאור	ערך
SAMPLING_DELAY_MS	השהייה בין דגימות	1000ms
SAMPLES_RANGE_TIME_MS	טווח זמן דגימה	6000ms
MAX_SAMPLES	מספר מקסימלי של דגימות	6
WIFI_TIMEOUT_MS	timeout חיבור WiFi	10000ms
HTTP_TIMEOUT_MS	timeout בקשות HTTP	5000ms

כתובות API

קבוע	ערך
SERVER_BASE_URL	"http://62.90.195.107"
DEVICES_LOCATION_ENDPOINT	"/api/v1/devices/location"
COMPRESSION_ENDPOINT	"/api/v1/compressions"
EMPTYING_EVENT_ENDPOINT	"/api/v1/emptying-events"

פונקציות עיקריות

main.cpp

פונקציה	תיאור
setup()	אתחול כל הרכיבים, טעינת קונפיגורציה, חיבור WiFi
loop()	לולאה ראשית, עדכון רכיבים, טיפול בטריגרים

compression_sensor.cpp

פונקציה	תיאור
begin()	אתחול חיישן, הגדרת פינים
update()	עדכון מצב דגימה, קריאת מתח
startSampling()	התחלת דגימת מתח
stopSampling()	עצירת דגימה, חישוב מקסימום
readVoltage()	קריאת מתח מהפין האנלוגי
getMaxVoltage()	החזרת ערך מתח מקסימלי
convertToVoltage()	המרת ערך ADC למתח (3.3V/4095)

wifi_manager.cpp

פונקציה	תיאור
begin()	אתחול WiFi, ניסיון חיבור
connect()	חיבור ל-WiFi עם timeout
startAPMode()	מעבר למצב Access Point
stopAPMode()	יציאה ממצב AP
isConnected()	בדיקת חיבור WiFi
updateLedStatus()	עדכון LED לפי מצב WiFi

trigger_manager.cpp

פונקציה	תיאור
`begin()`	אתחול פיני טריגר
`update()`	בדיקת שינויי מצב טריגר
`isStartTriggered()`	בדיקה אם טריגר התחלה פעיל
`isEventTriggered()`	בדיקה אם טריגר אירוע פעיל
`isWiFiResetPressed()`	בדיקה אם לחצן איפוס WiFi לחוץ

event_sender.cpp

פונקציה	תיאור
`sendCompressionEvent()`	שליחת אירוע דחיסה לשרת
`sendEmptyingEvent()`	שליחת אירוע פינוי לשרת
`sendLocationUpdate()`	שליחת עדכון מיקום לשרת
`retryUntilSuccess()`	ניסיון חוזר עד הצלחה

מצבי עבודה

מצב רגיל

- חיבור WiFi פעיל
- ניטור טריגרים
- דגימת מתח בעת דחיסה
- שליחת אירועים לשרת
- עדכון LED סטטוס

מצב AP (Access Point)

- יצירת רשת WiFi "ESP32_SmartCompactorLogge"
- שרת web 192.168.4.1 על כתובת
- טופס קונפיגורציה
- שמירת הגדרות חדשות
- חזרה למצב רגיל

מצב שגיאה

- ניסיון חוזר לחיבור WiFi
- שליחה חוזרת של אירועים
- לוגים מפורטים
- עדכון LED שגיאה

שרת Backend - קוד Python

מבנה קבצים

קבצים ראשיים

קובץ	תיאור
main.py	אפליקציית FastAPI ראשית
config.py	קונפיגורציה מרכזית
constants.py	קבועים ו-enums
lifespan.py	ניהול מחזור חיים
init.sql	סכמת מסד נתונים

תיקיות

תיקייה	תיאור
routers/	נתבי API
models/	מודלי ORM
schemas/	סכמות Pydantic
services/	לוגיקה עסקית
utils/	כלי עזר
db/	חיבור למסד נתונים
tasks/	משימות רקע
templates/	תבניות HTML

קבצים מפורטים

main.py

- יצירת FastAPI עם כותרת, גרסה, תיאור
- הוספת CORS - תמיכה בבקשות cross-origin
- חיבור routers - כל נתבי ה-API
- הגדרת templates - Jinja2 לתבניות HTML
- טיפול בשגיאות - global exception handler
- הרצת uvicorn - שרת ASGI

config.py

קלאס	תיאור
`Environment`	enum לסביבות (development, production, testing)
`DatabaseConfig`	הגדרות מסד נתונים (host, port, database, username, password)
`AppConfig`	הגדרות אפליקציה (title, version, CORS, API prefix, scheduler)
`Config`	wrapper לכל ההגדרות

constants.py

קבוע	תיאור
`ResponseStatus`	סטטוסי תגובה (success, error, warning)
`HTTPStatus`	קודי HTTP
`APIMessages`	הודעות API
`TableNames`	שמות טבלאות
`APIEndpoints`	נתיבי API
`ApiPaths`	נתיבים ראשיים
`ContentTypes`	סוגי תוכן
`DateFormats`	פורמטי תאריכים
`SchedulerConstants`	קבועי תזמון
`ValidationConstants`	קבועי ולידציה
`LoggingConstants`	קבועי לוגים
`ApiTags`	תגיות API

Routers (נתבי API)

compression_router.py

- `POST /compressions` - יצירת אירוע דחיסה
- `CompressionCreateRequest` - סכמת בקשה
- טיפול בשגיאות - ValueError (404), Exception (500)

emptying_router.py

- `POST /emptying-events` - יצירת אירוע פינוי
- `EmptyingEventCreateRequest` - סכמת בקשה
- טיפול בשגיאות - ValueError (404), Exception (500)

device_router.py

- `POST /devices/location` - עדכון מיקום מכשיר

- `GET /devices` - קבלת רשימת מכשירים עם דגימות
- `DeviceLocationUpdateRequest` - סכמת בקשה

daily_average.py

- `GET /daily-average` - קבלת ממוצעים יומיים
- `GET /daily-average/latest` - ממוצע אחרון
- `POST /daily-average/calculate` - חישוב ידני
- פאגינציה - skip, limit
- ולידציה - פורמט תאריך

health_router.py

- `GET /health` - בדיקת בריאות המערכת
- סטטוס DB - חיבור למסד נתונים
- סטטוס Scheduler - מצב התזמון

device_management.py

- `GET /devices/manage` - דף ניהול מכשירים
- `POST /devices/add` - הוספת מכשיר
- `POST /devices/edit` - עריכת מכשיר
- `POST /devices/delete` - מחיקת מכשיר

Models (מודלי ORM)

Device

- **שדות**: id, name, latitude, longitude, location_address, last_emptying_timestamp, compression_count, created_at, updated_at
- **פונקציות**: to_dict(), repr()

CompressionEvent

- **שדות**: id, device_id, timestamp, voltage_max, capacity_in_percent, created_at, updated_at
- **פונקציות**: to_dict(), repr()
- **יחסים**: ForeignKey ל-Device

EmptyingEvent

- **שדות**: id, device_id, timestamp, created_at, updated_at
- **פונקציות**: to_dict(), repr()

- **יחסים**: ForeignKey ל-Device

DailyAverage

- **שדות**: id, device_id, date, average, calculated_at, created_at, updated_at
- **פונקציות**: to_dict(), repr()
- **ייחודיות**: (device_id, date)

Schemas (סכמות Pydantic)

Device Schemas

סכמה	תיאור
DeviceResponse	תשובת מכשיר בודד
DeviceSampleResponse	דגימת דחיסה
DeviceWithSamplesResponse	מכשיר עם דגימות
DeviceLocationUpdateRequest	בקשת עדכון מיקום
DeviceCreateRequest	בקשת יצירת מכשיר

Compression Schemas

סכמה	תיאור
CompressionEventResponse	תשובת אירוע דחיסה
CompressionCreateRequest	בקשת יצירת אירוע דחיסה

Emptying Schemas

סכמה	תיאור
EmptyingEventResponse	תשובת אירוע פינוי
EmptyingEventCreateRequest	בקשת יצירת אירוע פינוי

Daily Average Schemas

סכמה	תיאור
DailyAverageResponse	תשובת ממוצע יומי

Services (לוגיקה עסקית)

CompressionService

פונקציה	תיאור
_calculate_capacity()	חישוב אחוז קיבולת לפי מתח או מונה דחיסות
create_compression_event()	יצירת אירוע דחיסה, עדכון מונה, חישוב קיבולת

EmptyingService

פונקציה	תיאור
create_emptying_event()	יצירת אירוע פינוי, איפוס מונה דחיסות, עדכון זמן פינוי

DeviceService

פונקציה	תיאור
get_all_devices()	קבלת כל המכשירים
get_device()	קבלת מכשיר לפי מזהה
create_device()	יצירת מכשיר חדש
update_device()	עדכון מכשיר
delete_device()	מחיקת מכשיר
get_or_create_device()	קבלת או יצירת מכשיר
update_device_location()	עדכון מיקום וכתובת
get_devices_with_recent_compressions()	מכשירים עם דגימות אחרונות

DailyAverageService

פונקציה	תיאור
create_daily_average()	יצירת או עדכון ממוצע יומי
get_daily_averages()	קבלת ממוצעים עם פאגינציה
get_latest_daily_average()	ממוצע אחרון
calculate_and_save_daily_averages()	חישוב ממוצעים לתאריך
calculate_yesterday_average()	חישוב ממוצע אתמול

Utils (כלי עזר)

timezone_utils.py

פונקציה	תיאור
to_israel()	המרה לאזור זמן ישראל
to_utc()	המרה ל-UTC
now_israel()	זמן נוכחי בישראל
parse_iso_datetime()	המרת מחרוזת ISO ל-datetime
serialize_with_israel_tz()	המרת אובייקטים לזמן ישראל

geocode_utils.py

פונקציה	תיאור
get_address_from_latlon()	המרת קואורדינטות לכתובת בעברית
	שימוש ב-Nominatim API - OpenStreetMap

Database (מסד נתונים)

database.py

רכיב	תיאור
engine	מנוע SQLAlchemy
SessionLocal	מפעל sessions
Base	בסיס למודלים
get_db()	dependency ל-FastAPI
create_tables()	יצירת טבלאות
drop_tables()	מחיקת טבלאות

Tasks (משימות רקע)

scheduler.py

רכיב	תיאור
scheduler	AsyncIOScheduler עם timezone ישראל
calculate_daily_average_job()	משימה לחישוב ממוצע יומי
setup_scheduler()	הגדרת תזמון
shutdown_scheduler()	עצירת תזמון

Lifespan (מחזור חיים)

lifespan.py

שלב	תיאור
Startup	יצירת טבלאות, התחלת תזמון
Shutdown	עצירת תזמון
	טיפול בשגיאות - לוגים מפורטים

תהליכי עבודה מפורטים

1. אתחול המערכת

בקר ESP32

```mermaid
graph TD
  A[טעינת קונפיגורציה] --> B[קריאה מהזיכרון]
  B --> C[אתחול רכיבים]
  C --> D[WiFi, חיישנים, טריגרים, LED]
  D --> E[חיבור WiFi]
  E --> F[ניסיון חיבור עם timeout]
  F --> G[סנכרון NTP]
  G --> H[קבלת זמן מדויק]
  H --> I[שליחת מיקום ראשוני]
  I --> J[לשרת עם ריפוי]
```

שרת Backend

```mermaid
graph TD
  A[טעינת קונפיגורציה] --> B[מהסביבה או ברירת מחדל]
  B --> C[חיבור למסד נתונים]
  C --> D[יצירת engine ו-sessions]
  D --> E[יצירת טבלאות]
  E --> F[אם לא קיימות]
  F --> G[התחלת תזמון]
  G --> H[APScheduler]
  H --> I[הפעלת FastAPI]
  I --> J[עם כל ה-routers]
```

2. תהליך דחיסה

זיהוי דחיסה

תיאור	שלב
לחצן או חיישן	**טריגר פיזי**
עלייה או ירידה במתח	**זיהוי edge**
מעבר למצב SAMPLING	**התחלת דגימה**

דגימת מתח

```mermaid
graph TD
    A[קריאת ADC] --> B[ערך גולמי 0-4095]
    B --> C[המרה למתח]
    C --> D[raw * 3.3 / 4095]
    D --> E[שמירה במערך]
    E --> F[עד MAX_SAMPLES]
    F --> G[חישוב מקסימום]
    G --> H[מעקב אחר ערך גבוה ביותר]
    H --> I[בדיקת סיום]
    I --> J[לפי זמן או מספר דגימות]
```

שליחה לשרת

```json
{
  "device_id": "ESP32_001",
  "timestamp": "2025-07-27T10:30:00Z",
  "voltage_max": 2.85
}
```

תהליך שליחה:

1. יצירת JSON - device_id, timestamp, voltage_max
2. שליחת HTTP POST - לכתובת `/api/v1/compressions`
3. בדיקת תשובה - קוד 200/201
4. ריפוי אוטומטי - ניסיון חוזר עד הצלחה

עיבוד בשרת

תיאור	שלב
בדיקת פורמט נתונים	**ולידציה**
לפי מתח או מונה דחיסות	**חישוב קיבולת**
טבלת compression_events	**שמירה במסד**
מונה דחיסות, זמן עדכון	**עדכון מכשיר**
אישור קבלה	**תשובה לבקר**

3. תהליך פינוי

זיהוי פינוי

```mermaid
graph TD
    A[טריגר פיזי] --> B[חיישן או לחצן]
    B --> C[יצירת אירוע]
    C --> D[נוכחי timestamp]
    D --> E[שליחה לשרת]
    E --> F[/api/v1/emptying-events]
```

עיבוד בשרת

פעולה	שלב
בדיקת device_id	**ולידציה**
טבלת emptying_events	**יצירת אירוע**
compression_count = 0	**איפוס מונה**
last_emptying_timestamp	**עדכון זמן**
אישור קבלה	**תשובה לבקר**

4. חישוב ממוצעים יומיים

תזמון אוטומטי

```mermaid
```

```mermaid
graph TD
  A[APScheduler] --> B[ריצה יומית בשעה קבועה]
  B --> C[חישוב אתמול]
  C --> D[לפי שעון ישראל]
  D --> E[איסוף נתונים]
  E --> F[כל אירועי הדחיסה של אתמול]
  F --> G[חישוב ממוצע]
  G --> H[סכום מתחים / מספר דגימות]
  H --> I[שמירה במסד]
  I --> J[טבלת daily_averages]
```

חישוב ידני

```json
{
  "date": "2025-07-26"
}
```

תהליך:

1. בקשת API - `POST /daily-average/calculate`
2. פורמט תאריך - YYYY-MM-DD ולידציה
3. איסוף נתונים - לפי תאריך
4. חישוב ממוצע - לכל מכשיר בנפרד
5. שמירה במסד - עם ייחודיות (device_id, date)

5. ניהול מכשירים

הוספת מכשיר

```mermaid
graph TD
  A[דף ניהול] --> B[טופס הוספה]
  B --> C[ולידציה]
  C --> D[מזהה ייחודי]
  D --> E[יצירה במסד]
  E --> F[טבלת devices]
  F --> G[תשובה]
  G --> H[אישור יצירה]
```

עדכון מיקום

בקשת API:

```json
{
  "device_id": "ESP32_001",
  "latitude": 32.0853,
  "longitude": 34.7818
}
```

תהליך:

1. גיאוקודינג - המרת קואורדינטות לכתובת
2. עדכון במסד - latitude, longitude, location_address
3. תשובה - נתוני מכשיר מעודכנים

מחיקת מכשיר

שלב	פעולה
בקשת מחיקה	POST /devices/delete
בדיקת תלויות	אירועי דחיסה/פינוי
מחיקה במסד	עם CASCADE או שגיאה
תשובה	אישור או שגיאה

ניהול שגיאות וריפוי

שגיאות בקר ESP32

שגיאות WiFi

שגיאה	טיפול
חיבור נכשל	ניסיון חוזר כל 30 שניות
איבוד חיבור	זיהוי אוטומטי, ניסיון חיבור מחדש
אין קונפיגורציה	מעבר למצב AP

שגיאות HTTP

שגיאה	טיפול
timeout	ניסיון חוזר עם השהייה
שגיאת שרת	ניסיון חוזר עד הצלחה
שגיאת JSON	לוג מפורט, ניסיון חוזר

שגיאות חיישן

שגיאה	טיפול
קריאה נכשלה	לוג שגיאה, המשך עבודה
ערך לא תקין	ולידציה, דחייה
חיישן לא זמין	מצב ERROR, ניסיון חוזר

שגיאות שרת

שגיאות מסד נתונים

שגיאה	טיפול
חיבור נכשל	retry עם backoff
שגיאת SQL	rollback, לוג מפורט
אינטגריטי	בדיקת foreign keys

שגיאות API

קוד	תיאור	טיפול
400	Bad Request	ולידציה עם פרטים
404	Not Found	לא נמצא
500	Internal Server Error	שגיאת שרת

שגיאות תזמון

שגיאה	טיפול
משימה נכשלה	לוג מפורט, המשך עבודה
תזמון נכשל	ניסיון חוזר
זמן לא תקין	המרה לאזור זמן נכון

מנגנוני ריפוי

ריפוי אוטומטי

```python
```

```python
def retry_with_backoff(func, max_retries=3):
    for attempt in range(max_retries):
        try:
            return func()
        except Exception as e:
            wait_time = 2 ** attempt  # Exponential backoff
            time.sleep(wait_time)
            if attempt == max_retries - 1:
                raise e
```

מאפיינים:

- ניסיון חוזר - עם השהייה הולכת וגדלה
- מקסימום ניסיונות - הגבלה למניעת לולאה אינסופית
- לוגים מפורטים - לכל ניסיון ושגיאה

מצבי חירום

מצב	תיאור
מצב AP	כאשר אין WiFi
מצב offline	שמירה מקומית עד חיבור
איפוס מלא	לחצן פיזי לאיפוס הגדרות

ממשקי משתמש

דף ניהול מכשירים

עיצוב

```css
```

```
:root {
  --primary-color: #2563eb;
  --secondary-color: #64748b;
  --success-color: #059669;
  --danger-color: #dc2626;
  --gradient: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
}

.container {
  max-width: 1200px;
  margin: 0 auto;
  padding: 2rem;
  direction: rtl;
}

.card {
  background: white;
  border-radius: 12px;
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
  padding: 2rem;
  margin-bottom: 2rem;
}
```

מאפיינים:

- CSS Variables - צבעים, gradients, shadows
- תמיכה RTL - עברית מימין לשמאל
- רספונסיבי - התאמה לכל גודל מסך
- אנימציות - transitions, hover effects

פונקציונליות

תכונה	תיאור
טבלת מכשירים	רשימה עם מיון וחיפוש
טופס הוספה	מזהה, שם, מיקום
טופס עריכה	עדכון פרטי מכשיר
מחיקה	עם אישור
צפייה בנתונים	היסטוריית דחיסות/פינויים

JavaScript

```javascript
```

```
// AJAX request example
async function addDevice(deviceData) {
    try {
        const response = await fetch('/devices/add', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify(deviceData)
        });

        if (response.ok) {
            showSuccess('מכשיר נוסף בהצלחה');
            refreshDeviceTable();
        } else {
            showError('שגיאה בהוספת המכשיר');
        }
    } catch (error) {
        showError('שגיאת רשת');
    }
}
```

מאפיינים:

- שליחת בקשות ללא רענון - AJAX requests
- ולידציה - בדיקת טופס בצד לקוח
- עדכון דינמי - טבלה מתעדכנת אוטומטית
- הודעות - success/error messages

API Documentation (Swagger)

תכונות

תכונה	תיאור
כתובת	http://62.90.195.107/docs
ניסוי API	שליחת בקשות ישירות מהדפדפן
דוגמאות	לכל endpoint
סכמות	Pydantic models
תגובות	קודי HTTP ודוגמאות

תזמון ומשימות רקע

APScheduler

הגדרות

```python
from apscheduler.schedulers.asyncio import AsyncIOScheduler
from pytz import timezone

scheduler = AsyncIOScheduler(
    timezone=timezone('Asia/Jerusalem'),
    job_defaults={
        'coalesce': True,  # מיזוג משימות זהות
        'max_instances': 1  # 1 מקסימום instance למשימה
    }
)
```

משימות

משימה	תזמון	תיאור
חישוב ממוצע יומי	יומי 00:30	ריצה יומית בשעה קבועה
ניקוי לוגים	שבועי	מחיקת לוגים ישנים
גיבוי נתונים	יומי 02:00	שמירת גיבוי אוטומטי

חישוב ממוצעים

תהליך

```python
```

```python
async def calculate_daily_average_job():
    try:
        yesterday = (datetime.now(timezone('Asia/Jerusalem')) - timedelta(days=1)).date()

        # איסוף נתונים
        compression_events = db.query(CompressionEvent).filter(
            func.date(CompressionEvent.timestamp) == yesterday
        ).all()

        # חישוב ממוצע לכל מכשיר
        device_averages = {}
        for event in compression_events:
            if event.device_id not in device_averages:
                device_averages[event.device_id] = []
            device_averages[event.device_id].append(event.voltage_max)

        # שמירה במסד
        for device_id, voltages in device_averages.items():
            average = sum(voltages) / len(voltages)
            daily_average = DailyAverageService.create_daily_average(
                device_id=device_id,
                date=yesterday,
                average=average
            )

        logger.info(f"חושבו ממוצעים יומיים עבור {len(device_averages)} מכשירים")

    except Exception as e:
        logger.error(f"שגיאה בחישוב ממוצעים יומיים: {str(e)}")
```

טיפול בשגיאות

מצב	טיפול
אין נתונים	לוג warning, המשך עבודה
שגיאת DB	rollback, ניסיון חוזר
שגיאת חישוב	לוג error, המשך עבודה

קונפיגורציה והגדרות

בקר ESP32

קובץ config.h

```cpp
```

```
// פינים (Pins)
#define TRIGGER_PIN_START 15    // פין דיגיטלי לטריגר התחלה
#define TRIGGER_PIN_EVENT 4     // פין דיגיטלי לטריגר אירוע
#define VOLTAGE_PIN 34          // פין אנלוגי לקריאת מתח
#define LED_PIN 2               // מובנה לסטטוס LED
#define WIFI_RESET_PIN 16       // פין לאיפוס WiFi

// זמנים (Timing)
#define SAMPLING_DELAY_MS 1000       // השהייה בין דגימות (1 שנייה)
#define SAMPLES_RANGE_TIME_MS 6000   // טווח זמן דגימה
#define MAX_SAMPLES 6                // מספר מקסימלי של דגימות
#define WIFI_TIMEOUT_MS 10000        // timeout חיבור WiFi
#define HTTP_TIMEOUT_MS 5000         // timeout בקשות HTTP

// כתובות API
#define SERVER_BASE_URL "http://62.90.195.107"
#define DEVICES_LOCATION_ENDPOINT "/api/v1/devices/location"
#define COMPRESSION_ENDPOINT "/api/v1/compressions"
#define EMPTYING_EVENT_ENDPOINT "/api/v1/emptying-events"
```

קובץ config.cpp

```cpp
#include "config.h"

// הגדרות ברירת מחדל
const char* DEFAULT_WIFI_SSID = "";
const char* DEFAULT_WIFI_PASSWORD = "";
const char* DEFAULT_DEVICE_ID = "ESP32_DEFAULT";
const char* DEFAULT_SERVER_URL = SERVER_BASE_URL;

// הודעות מערכת
const char* MSG_WIFI_CONNECTING = "מתחבר ל-WiFi...";
const char* MSG_WIFI_CONNECTED = "מחובר WiFi";
const char* MSG_WIFI_FAILED = "נכשל חיבור WiFi";
const char* MSG_SAMPLING_START = "התחלת דגימה";
const char* MSG_SAMPLING_STOP = "סיום דגימה";
const char* MSG_EVENT_SENT = "אירוע נשלח";
const char* MSG_EVENT_FAILED = "שליחת אירוע נכשלה";
```

שרת Backend

קובץ config.py

```python
```

```python
from enum import Enum
from typing import List
import os

class Environment(str, Enum):
    DEVELOPMENT = "development"
    PRODUCTION = "production"
    TESTING = "testing"

class DatabaseConfig:
    def __init__(self):
        self.host = os.getenv("DB_HOST", "localhost")
        self.port = int(os.getenv("DB_PORT", "5432"))
        self.database = os.getenv("DB_NAME", "smart_compactor")
        self.username = os.getenv("DB_USER", "postgres")
        self.password = os.getenv("DB_PASSWORD", "password")

    @property
    def url(self) -> str:
        return f"postgresql://{self.username}:{self.password}@{self.host}:{self.port}/{self.database}"

class AppConfig:
    def __init__(self):
        self.title = "SmartCompactorLogger API"
        self.version = "1.0.0"
        self.description = "API למערכת ניטור פחי אשפה חכמים"
        self.cors_origins: List[str] = ["*"]
        self.api_prefix = "/api/v1"
        self.scheduler_timezone = "Asia/Jerusalem"

class Config:
    def __init__(self):
        self.environment = Environment(os.getenv("ENVIRONMENT", "development"))
        self.database = DatabaseConfig()
        self.app = AppConfig()
        self.debug = self.environment == Environment.DEVELOPMENT
```

קובץ constants.py

```python
```

```python
from enum import Enum

class ResponseStatus(str, Enum):
    SUCCESS = "success"
    ERROR = "error"
    WARNING = "warning"

class HTTPStatus(int, Enum):
    OK = 200
    CREATED = 201
    BAD_REQUEST = 400
    NOT_FOUND = 404
    INTERNAL_SERVER_ERROR = 500

class APIMessages:
    DEVICE_CREATED = "מכשיר נוצר בהצלחה"
    DEVICE_UPDATED = "מכשיר עודכן בהצלחה"
    DEVICE_DELETED = "מכשיר נמחק בהצלחה"
    DEVICE_NOT_FOUND = "מכשיר לא נמצא"
    COMPRESSION_CREATED = "אירוע דחיסה נוצר בהצלחה"
    EMPTYING_CREATED = "אירוע פינוי נוצר בהצלחה"
    DAILY_AVERAGE_CALCULATED = "ממוצע יומי חושב בהצלחה"

class TableNames:
    DEVICES = "devices"
    COMPRESSION_EVENTS = "compression_events"
    EMPTYING_EVENTS = "emptying_events"
    DAILY_AVERAGES = "daily_averages"

class APIEndpoints:
    HEALTH = "/health"
    DEVICES = "/devices"
    DEVICES_LOCATION = "/devices/location"
    COMPRESSIONS = "/compressions"
    EMPTYING_EVENTS = "/emptying-events"
    DAILY_AVERAGE = "/daily-average"
    DAILY_AVERAGE_LATEST = "/daily-average/latest"
    DAILY_AVERAGE_CALCULATE = "/daily-average/calculate"

class ApiPaths:
    API_V1 = "/api/v1"
    DOCS = "/docs"
    REDOC = "/redoc"

class ContentTypes:
    JSON = "application/json"
```

```
    HTML = "text/html"
    CSS = "text/css"
    JAVASCRIPT = "application/javascript"

class DateFormats:
    ISO_DATE = "%Y-%m-%d"
    ISO_DATETIME = "%Y-%m-%dT%H:%M:%SZ"
    DISPLAY_DATE = "%d/%m/%Y"
    DISPLAY_DATETIME = "%d/%m/%Y %H:%M"

class SchedulerConstants:
    DAILY_AVERAGE_JOB_ID = "calculate_daily_average"
    DAILY_AVERAGE_HOUR = 0
    DAILY_AVERAGE_MINUTE = 30
    CLEANUP_JOB_ID = "cleanup_old_logs"
    BACKUP_JOB_ID = "backup_database"

class ValidationConstants:
    MIN_LATITUDE = -90.0
    MAX_LATITUDE = 90.0
    MIN_LONGITUDE = -180.0
    MAX_LONGITUDE = 180.0
    MIN_VOLTAGE = 0.0
    MAX_VOLTAGE = 5.0
    MAX_DEVICE_ID_LENGTH = 64
    MAX_DEVICE_NAME_LENGTH = 128

class LoggingConstants:
    FORMAT = "%(asctime)s - %(name)s - %(levelname)s - %(message)s"
    DATE_FORMAT = "%Y-%m-%d %H:%M:%S"
    MAX_LOG_SIZE = 10 * 1024 * 1024  # 10MB
    BACKUP_COUNT = 5

class ApiTags:
    DEVICES = "מכשירים"
    COMPRESSIONS = "דחיסות"
    EMPTYING = "פינויים"
    DAILY_AVERAGES = "ממוצעים יומיים"
    HEALTH = "בריאות המערכת"
    MANAGEMENT = "ניהול"
```

משתני סביבה

.env (שרת)

```bash
```

```
# Database Configuration
DB_HOST=localhost
DB_PORT=5432
DB_NAME=smart_compactor
DB_USER=postgres
DB_PASSWORD=your_secure_password

# Application Configuration
ENVIRONMENT=production
DEBUG=false
API_PREFIX=/api/v1

# Server Configuration
HOST=0.0.0.0
PORT=8000

# Security
SECRET_KEY=your_super_secret_key_here
CORS_ORIGINS=["http://localhost:3000", "https://yourdomain.com"]

# External Services
NOMINATIM_USER_AGENT=SmartCompactorLogger/1.0

# Logging
LOG_LEVEL=INFO
LOG_FILE=/var/log/smart_compactor.log

# Scheduler
SCHEDULER_TIMEZONE=Asia/Jerusalem
DAILY_CALCULATION_HOUR=0
DAILY_CALCULATION_MINUTE=30
```

קונפיגורציה בקר

פרמטרי WiFi וחיבור

פרמטר	תיאור	דוגמה
WiFi SSID	שם הרשת	"MyNetwork"
WiFi Password	סיסמת הרשת	"MySecurePassword123"
Device ID	מזהה ייחודי של המכשיר	"ESP32_YARD_001"
Server URL	כתובת השרת	"http://62.90.195.107"
API Endpoints	נתיבי API	"/api/v1/compressions"

דוגמאות שימוש

1. הגדרת מכשיר חדש

שלב 1: הגדרת בקר ESP32

```cpp
// config.h - הגדרות מכשיר ספציפי
#define DEVICE_ID "ESP32_PARK_MAIN_001"
#define WIFI_SSID "CityWiFi"
#define WIFI_PASSWORD "SecurePassword123"

// במקרה של מכשיר ללא קונפיגורציה
// עם הרשת AP הבקר יעבור למצב:
// SSID: "ESP32_SmartCompactorLogger"
// IP: 192.168.4.1
```

שלב 2: רישום במערכת

HTTP Request:

```http
POST /api/v1/devices/location
Content-Type: application/json

{
  "device_id": "ESP32_PARK_MAIN_001",
  "latitude": 32.0853,
  "longitude": 34.7818
}
```

Response:

```json
```

```
{
  "id": "ESP32_PARK_MAIN_001",
  "name": null,
  "latitude": 32.0853,
  "longitude": 34.7818,
  "location_address": "רחוב דיזנגוף 1, תל אביב",
  "last_emptying_timestamp": null,
  "compression_count": 0,
  "created_at": "2025-07-27T10:00:00Z",
  "updated_at": "2025-07-27T10:00:00Z"
}
```

2. תהליך דחיסה מלא

שלב 1: זיהוי דחיסה בבקר

```cpp
void handleCompressionTrigger() {
    Serial.println("זוהתה דחיסה - מתחיל דגימה");

    // התחלת דגימת מתח
    compressionSensor.startSampling();

    // המתנה לסיום דגימה
    while (compressionSensor.isSampling()) {
        compressionSensor.update();
        delay(100);
    }

    // קבלת מתח מקסימלי
    float maxVoltage = compressionSensor.getMaxVoltage();
    Serial.printf("מתח מקסימלי נמדד: %.2fV\n", maxVoltage);

    // שליחה לשרת
    eventSender.sendCompressionEvent(maxVoltage);
}
```

שלב 2: עיבוד בשרת

```python
```

```python
@router.post("/compressions")
async def create_compression_event(
    request: CompressionCreateRequest,
    db: Session = Depends(get_db)
):
    try:
        # יצירת אירוע דחיסה
        compression_event = await CompressionService.create_compression_event(
            db=db,
            device_id=request.device_id,
            timestamp=request.timestamp,
            voltage_max=request.voltage_max
        )

        logger.info(f"נוצר אירוע דחיסה למכשיר {request.device_id}")

        return CompressionEventResponse.from_orm(compression_event)

    except ValueError as e:
        raise HTTPException(status_code=404, detail=str(e))
    except Exception as e:
        logger.error(f"שגיאה ביצירת אירוע דחיסה: {str(e)}")
        raise HTTPException(status_code=500, detail="שגיאה פנימית")
```

שלב 3: חישוב קיבולת

```python
def _calculate_capacity(self, device: Device, voltage_max: float) -> float:
    """חישוב אחוז קיבולת על בסיס מתח ומונה דחיסות"""

    # חישוב על בסיס מתח (0V = 0%, 3.3V = 100%)
    voltage_percentage = min((voltage_max / 3.3) * 100, 100)

    # חישוב על בסיס מונה דחיסות (עד 20 דחיסות = 100%)
    compression_percentage = min((device.compression_count / 20) * 100, 100)

    # ממוצע משוקלל (70% מתח, 30% מונה)
    capacity = (voltage_percentage * 0.7) + (compression_percentage * 0.3)

    return round(capacity, 2)
```

3. אירוע פינוי

בקר ESP32

```cpp
void handleEmptyingEvent() {
    Serial.println("זוהה אירוע פינוי");

    // שליחת אירוע פינוי לשרת
    bool success = eventSender.sendEmptyingEvent();

    if (success) {
        Serial.println("אירוע פינוי נשלח בהצלחה");
        ledManager.showSuccess();
    } else {
        Serial.println("שגיאה בשליחת אירוע פינוי");
        ledManager.showError();
    }
}
```

שרת

```python
@router.post("/emptying-events")
async def create_emptying_event(
    request: EmptyingEventCreateRequest,
    db: Session = Depends(get_db)
):
    try:
        # יצירת אירוע פינוי
        emptying_event = await EmptyingService.create_emptying_event(
            db=db,
            device_id=request.device_id,
            timestamp=request.timestamp
        )

        logger.info(f"נוצר אירוע פינוי למכשיר {request.device_id}")

        return EmptyingEventResponse.from_orm(emptying_event)

    except ValueError as e:
        raise HTTPException(status_code=404, detail=str(e))
```

4. ניהול מכשיר בממשק Web

HTML Form

```html
```

```html
<form id="deviceForm" class="device-form">
  <h3>הוספת מכשיר חדש</h3>

  <div class="form-group">
    <label for="deviceId">מזהה מכשיר:</label>
    <input type="text" id="deviceId" name="device_id" required
        placeholder="ESP32_LOCATION_001">
  </div>

  <div class="form-group">
    <label for="deviceName">שם מכשיר:</label>
    <input type="text" id="deviceName" name="name"
        placeholder="פח ראשי - פארק הירקון">
  </div>

  <div class="form-row">
    <div class="form-group">
      <label for="latitude">קו רוחב:</label>
      <input type="number" id="latitude" name="latitude"
          step="0.000001" min="-90" max="90" required>
    </div>

    <div class="form-group">
      <label for="longitude">קו אורך:</label>
      <input type="number" id="longitude" name="longitude"
          step="0.000001" min="-180" max="180" required>
    </div>
  </div>

  <button type="submit" class="btn btn-primary">הוסף מכשיר</button>
</form>
```

JavaScript Handler

```javascript
```

```
document.getElementById('deviceForm').addEventListener('submit', async (e) => {
    e.preventDefault();

    const formData = new FormData(e.target);
    const deviceData = Object.fromEntries(formData);

    try {
        const response = await fetch('/devices/add', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify(deviceData)
        });

        if (response.ok) {
            const result = await response.json();
            showNotification('מכשיר נוסף בהצלחה!', 'success');
            refreshDeviceTable();
            e.target.reset();
        } else {
            const error = await response.json();
            showNotification(`שגיאה: ${error.detail}`, 'error');
        }
    } catch (error) {
        showNotification('שגיאה בחיבור לשרת', 'error');
    }
});
```

5. שאילתות נתונים מתקדמות

קבלת נתוני מכשיר עם היסטוריה

```http
GET /api/v1/devices
```

```json
```

```
[
  {
    "device": {
      "id": "ESP32_PARK_MAIN_001",
      "name": "פח ראשי - פארק הירקון",
      "latitude": 32.0853,
      "longitude": 34.7818,
      "location_address": "פארק הירקון, תל אביב",
      "last_emptying_timestamp": "2025-07-26T14:30:00Z",
      "compression_count": 12,
      "created_at": "2025-07-20T10:00:00Z",
      "updated_at": "2025-07-27T10:15:00Z"
    },
    "recent_compressions": [
      {
        "id": 1520,
        "timestamp": "2025-07-27T10:15:00Z",
        "voltage_max": 2.85,
        "capacity_in_percent": 78.5
      },
      {
        "id": 1519,
        "timestamp": "2025-07-27T09:45:00Z",
        "voltage_max": 2.72,
        "capacity_in_percent": 75.2
      }
    ]
  }
]
```


חישוב ממוצע יומי

http

```
POST /api/v1/daily-average/calculate
Content-Type: application/x-www-form-urlencoded

date=2025-07-26
```

json

```
{
  "message": "ממוצע יומי חושב בהצלחה",
  "date": "2025-07-26",
  "devices_processed": 3,
  "averages_calculated": [
    {
      "device_id": "ESP32_PARK_MAIN_001",
      "average": 2.67
    },
    {
      "device_id": "ESP32_MALL_ENTRANCE_002",
      "average": 2.41
    },
    {
      "device_id": "ESP32_BEACH_SOUTH_003",
      "average": 2.89
    }
  ]
}
```


6. מצבי חירום וטיפול בשגיאות

איפוס קונפיגורציה ESP32

```cpp
void handleWiFiReset() {
  Serial.println("מתחיל איפוס WiFi...");

  // מחיקת קונפיגורציה שמורה
  configManager.clearWiFiConfig();

  // מעבר למצב AP
  wifiManager.startAPMode();

  // הודעה למשתמש
  Serial.println("מצב קונפיגורציה פעיל:");
  Serial.println("SSID: ESP32_SmartCompactorLogger");
  Serial.println("IP: 192.168.4.1");

  // הפעלת שרת קונפיגורציה
  webConfigServer.start();
}
```

דף קונפיגורציה Web

html

```html
<!DOCTYPE html>
<html dir="rtl">
<head>
  <meta charset="UTF-8">
  <title>קונפיגורציה - SmartCompactorLogger</title>
  <style>
    body { font-family: Arial, sans-serif; margin: 40px; }
    .container { max-width: 500px; margin: 0 auto; }
    .form-group { margin-bottom: 20px; }
    label { display: block; margin-bottom: 5px; font-weight: bold; }
    input { width: 100%; padding: 10px; border: 1px solid #ccc; border-radius: 5px; }
    button { background: #007bff; color: white; padding: 15px 30px; border: none; border-radius: 5px; cursor: pointer; }
    button:hover { background: #0056b3; }
  </style>
</head>
<body>
  <div class="container">
    <h1>קונפיגורציה - SmartCompactorLogger</h1>

    <form id="configForm">
      <div class="form-group">
        <label for="wifi_ssid">שם רשת WiFi:</label>
        <input type="text" id="wifi_ssid" name="wifi_ssid" required>
      </div>

      <div class="form-group">
        <label for="wifi_password">סיסמת WiFi:</label>
        <input type="password" id="wifi_password" name="wifi_password" required>
      </div>

      <div class="form-group">
        <label for="device_id">מזהה מכשיר:</label>
        <input type="text" id="device_id" name="device_id" required
           placeholder="ESP32_LOCATION_001">
      </div>

      <div class="form-group">
        <label for="server_url">כתובת שרת:</label>
        <input type="url" id="server_url" name="server_url"
           value="http://62.90.195.107" required>
      </div>

      <button type="submit">שמור הגדרות</button>
    </form>
  </div>
```

```
  <script>
    document.getElementById('configForm').addEventListener('submit', async (e) => {
      e.preventDefault();

      const formData = new FormData(e.target);
      const config = Object.fromEntries(formData);

      try {
        const response = await fetch('/save-config', {
          method: 'POST',
          headers: { 'Content-Type': 'application/json' },
          body: JSON.stringify(config)
        });

        if (response.ok) {
          alert('הגדרות נשמרו בהצלחה! המכשיר יאותחל מחדש...');
        } else {
          alert('שגיאה בשמירת ההגדרות');
        }
      } catch (error) {
        alert('שגיאה בחיבור');
      }
    });
  </script>
</body>
</html>
```

7. ניטור ובקרת איכות

דוח יומי אוטומטי

```python
```

```python
async def generate_daily_report():
    """יצירת דוח יומי עם סטטיסטיקות"""

    yesterday = datetime.now().date() - timedelta(days=1)

    # איסוף נתונים
    total_compressions = db.query(CompressionEvent).filter(
        func.date(CompressionEvent.timestamp) == yesterday
    ).count()

    total_emptyings = db.query(EmptyingEvent).filter(
        func.date(EmptyingEvent.timestamp) == yesterday
    ).count()

    active_devices = db.query(Device).filter(
        Device.updated_at >= yesterday
    ).count()

    # חישוב ממוצע עומס
    average_capacity = db.query(func.avg(CompressionEvent.capacity_in_percent)).filter(
        func.date(CompressionEvent.timestamp) == yesterday
    ).scalar() or 0

    # יצירת דוח
    report = {
        "date": yesterday.isoformat(),
        "statistics": {
            "total_compressions": total_compressions,
            "total_emptyings": total_emptyings,
            "active_devices": active_devices,
            "average_capacity": round(average_capacity, 2)
        },
        "recommendations": []
    }

    # המלצות אוטומטיות
    if average_capacity > 80:
        report["recommendations"].append("רמת מילוי גבוהה - מומלץ להגביר תדירות פינוי")

    if total_emptyings == 0:
        report["recommendations"].append("לא בוצעו פינויים - בדיקת תקינות מערכת נדרשת")

    return report
```

סיכום

מספקת פתרון מלא וחכם לניטור פחי אשפה עם דחסן חשמלי. המערכת **SmartCompactorLogger** מערכת מאפשרת:

יתרונות עיקריים

יתרון	תיאור
ניטור בזמן אמת	מעקב מתמיד אחר מצב הפחים
חיסכון במשאבים	אופטימיזציה של מסלולי פינוי
תחזוקה חכמה	זיהוי מוקדם של תקלות
ניתוח נתונים	דוחות ותחזיות מתקדמות
ממשק ידידותי	ניהול פשוט ואינטואיטיבי

טכנולוגיות מובילות

- **IoT** - אינטרנט הדברים עם ESP32
- **Cloud Computing** - עיבוד נתונים בענן
- **Real-time Analytics** - ניתוח בזמן אמת
- **RESTful API** - ממשק תקשורת סטנדרטי
- **Responsive Design** - ממשק מותאם לכל מכשיר

קנה מידה ועתיד

המערכת תוכננה להתרחבות ותומכת ב:

- **מאות מכשירים** במקביל
- **הרחבות עתידיות** - חיישנים נוספים
- **אינטגרציה** עם מערכות ניהול עירוניות
- **למידת מכונה** לתחזיות מתקדמות

תיעוד זה כולל את כל המידע הנדרש להתקנה, תפעול ותחזוקה של המערכת.