Live Streaming and Presentation of Sensors’ Data

Μοίρας Αλέξανδρος   
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών  
Εθνικό Μετσόβιο ΠολυτεχνείοΑθήνα, Ελλάδα  
el18081@mail.ntua.gr

Παντελαίος Δημήτριος   
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών  
Εθνικό Μετσόβιο ΠολυτεχνείοΑθήνα, Ελλάδα  
el18049@mail.ntua.gr

Κώδικας Εργασίας:

<https://github.com/dpantelaios/Analysis-and-Design-of-Information-Systems>

*Abstract*—Tα τελευταία χρόνια αυξάνονται με ραγδαίους ρυθμούς οι ηλεκτρονικές συσκευές που επιτρέπουν την σύνδεση και την ανταλλαγή δεδομένων. Για αυτό το λόγο οδηγηθήκαμε στην δημιουργία του Διαδικτύου των πραγμάτων ή αλλιώς Internet of Things(IoT), το οποίο αποτελεί ένα δίκτυο επικοινωνίας πληθώρας συσκευών και παρέχει την δυνατότητα λήψης, επεξεργασίας και απεικόνισης δεδομένων που παράγουν, αλλά και διαχείρισης της λειτουργίας διαφόρων συσκευών[1].

Keywords—Apache Kafka, Kafka streams, telegraf, grafana, influxdb, websockets, IoT, sensors

# Introduction

Σκοπός της εργασίας είναι η δημιουργία μιας εφαρμογής ΙοΤ, κατά την οποία θα παράγουμε δεδομένα από αισθητήρες, θα λαμβάνονται από κάποιους κόμβους όπου θα γίνεται η απαιτούμενη επεξεργασία τους και με την σειρά τους τα επεξεργασμένα δεδομένα θα αποστέλλονται στον αντίστοιχο κόμβο, όπου θα πραγματοποιούνται η αποθήκευση τους σε μια timeseries database και η αποστολή τους για live παρουσίαση στον κατάλληλο κόμβο.

# STEPS TO SET UP SYSTEm

Αρχικά κάνουμε git clone το repository που έχει παρατεθεί παραπάνω και περιλαμβάνει τον κώδικα της εργασίας, προκειμένου να τον κατεβάσουμε στο σύστημα από το οποίο θα γίνει η εκτέλεση του. Εναλλακτικά μπορούμε να μεταφερθούμε στο repository που περιέχει τον κώδικα και να τον κατεβάσουμε χειροκίνητα πατώντας Code Download Zip και στην συνέχεια να τον κάνουμε extract στο σύστημα που έγινε η αποθήκευση.

To project εκτελείται αποκλειστικά σε docker container, οπότε η μοναδική εγκατάσταση λογισμικού που θα πρέπει να κάνουμε είναι η εγκατάσταση της πλατφόρμας docker Desktop από [docker Desktop Download](https://www.docker.com/products/docker-desktop/).

Στην συνέχεια, προκειμένου να θέσουμε σε λειτουργία όλα τα docker container που χρησιμοποιεί η εφαρμογή και βρίσκεται ένα σε κάθε φάκελο του κώδικα που κατεβάσαμε στο πρώτο βήμα, πρέπει να ακολουθηθούν με την σειρά τα ακόλουθα βήματα:

* cd Messaging\_Broker\_Layer
* docker compose up
* cd ../ Presentation Layer
* docker compose up
* cd ../ Data-Storage Layer
* docker compose up
* cd ../ Live Streaming Layer Java
* docker compose build
* docker compose up
* cd ../Device Layer
* docker compose build
* docker compose up

Στην συνέχεια μπορούμε να δούμε την live απεικόνιση των δεδομένων πατώντας στο link:

<http://localhost:3000/d/grS74cbVz/live-streaming?orgId=1>

Τέλος μπορούμε να δούμε την απεικόνιση των δεδομένων που είναι αποθηκευμένα στην βάση πατώντας το link:

<http://localhost:3000/d/1Dr8HObVk/main?orgId=1>

# COMPONENT, SOFTWARE

Η εφαρμογή αποτελείται από 8 components τα οποία παρουσιάζονται παρακάτω:

1. Το component device\_layer είναι υπεύθυνο για την παραγωγή δεδομένων ανά 15 λεπτά για 7 αισθητήρες, ανά ημέρα για 2 αισθητήρες, 4-6 δεδομένα την ημέρα τυχαία επιλεγμένα για τον αισθητήρα ανίχνευσης κίνησης, καθώς και ετεροχρονισμένα δεδομένα για τον αισθητήρα μέτρησης κατανάλωσης νερού ανά 15 λεπτά. Τα ετεροχρονισμένα δεδομένα που παράγονται ανά 6 ώρες και αντιστοιχούν σε ημερομηνία 2 μέρες πίσω θεωρούνται late processed, ενώ αυτά που παράγονται ανά 30 ώρες και αντιστοιχούν σε 10 μέρες πίσω θεωρούνται late rejected.

Oι αισθητήρες που προσομοιώθηκαν τα αντίστοιχα όρια των πιθανών τιμών τους παρουσιάζονται παρακάτω:

TABLE I . DEVICES

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sensor | Sensor Name | Interval | Value Type | Value Range |
| Θερμοκρασία | TH1 | 15 min | float | 12-35 |
| Θερμοκρασία | TH2 | 15 min | float | 12-35 |
| Ενέργεια κλιματιστικών | HVAC1 | 15 min | float | 0-100 |
| Ενέργεια κλιματιστικών | HVAC2 | 15 min | float | 0-200 |
| Ενέργειας υπόλοιπων ηλεκτρικών συσκευών | MiAC1 | 15 min | float | 0-150 |
| Ενέργειας υπόλοιπων ηλεκτρικών συσκευών | MiAC2 | 15 min | float | 0-200 |
| Κατανάλωση νερού | W1 | 15 min | float | 0-1 |
| Αθροιστική ενέργεια | Etot | 1 day | float | 2600x24 – 1000 – 2600x24 + 1000 |
| Αθροιστική κατανάλωση νερού | Wtot | 1 day | float | 100-120 |
| Ανίχνευση κίνησης | Mov1 | - | int | 0-1 |

Fig1: Αισθητήρες που προσομοιώθηκαν

Tα δεδομένα που παράγονται αποστέλλονται στα αντίστοιχα topic που παρουσιάζονται στον παρακάτω πίνακα:

TABLE ΙΙ . ΤΟPICS

|  |  |
| --- | --- |
| Topic | Αισθητήρες |
| 15min | Αισθητήρες με interval 15 λεπτών, μαζί με τα ετεροχρονισμένα δεδομένα του αισθητήρα κατανάλωσης νερού |
| day | Αισθητήρες με interval 1 μέρα |
| movementSensor | Αισθητήρας ανίχνευσης κίνησης |

Fig2:Τα topic που στέλνονται τα δεδομένα κάθε sensor

1. Tα component kafka και zookeeper χρησιμοποιούνται για την επικοινωνία μεταξύ των services. Mε το component kafka-init δημιουργούμε topics με συγκεκριμένα ονόματα στα οποία τα services μπορούν να γράφουν και να διαβάζουν δεδομένα. Ακόμη το kafka χρησιμοποιείται για την αποθήκευση κάποιων δεδομένων κατά την επεξεργασία τους, η οποία θα αναλυθεί σε μετέπειτα component.
2. To component live\_streaming\_layer java διαβάζει από τα δεδομένα των αισθητήρων από τα αντίστοιχα topic και πραγματοποιεί την απαιτούμενη επεξεργασία σε κάθε αισθητήρα.

TABLE III AGGREGATIONS

|  |  |
| --- | --- |
| Αισθητήρας | Είδος επεξεργασίας |
| TH1 | Μέσος όρος τη μέρα |
| TH2 | Μέσος όρος τη μέρα |
| HVAC1 | Άθροισμα τη μέρα |
| HVAC2 | Άθροισμα τη μέρα |
| MiAC1 | Άθροισμα τη μέρα |
| MiAC2 | Άθροισμα τη μέρα |
| W1 | Άθροισμα τη μέρα συμπεριλαμβανομένων των late processed events |
| Etot | Μέγιστο κάθε μέρα |
| Wtot | Μέγιστο κάθε μέρα |
| Mov1 | Συνολικό άθροισμα |
| Etot | Διαφορά μεταξύ διαδοχικών ημερών |
| Wtot | Διαφορά μεταξύ διαδοχικών ημερών |
| Hvac1, Hvac2, MiAC1, MiAC1, Etot | Etot – (Hvac1\_daily\_sum + Hvac2\_daily\_sum + MiAC1\_daily\_sum + MiAC1\_daily\_sum) |
| W1, Wtot | Wtot – W1\_daily\_sum |

Fig3:Τύπος επεξεργασίας για κάθε αισθητήρα

Στην συνέχεια τόσο τα αρχικά δεδομένα όσο και αυτά που προήλθαν από επεξεργασία αποστέλλονται στον kafka στα αντίστοιχα topics που παρουσιάζονται παρακάτω:

TABLE IV TOPICS

|  |  |
| --- | --- |
| Topic | Δεδομένα |
| raw | Oλα τα δεδομένα 15λεπτων αισθητήρων πριν την επεξεργασία |
| aggDay15min | Oλα τα δεδομένα 15λεπτων αισθητήρων μετά την επεξεργασία (μέσος όρος ή άθροισμα) |
| aggDayDiff | Oλα τα δεδομένα ημερήσιων αισθητήρων μετά την επεξεργασία (διαφορά μεταξύ διαδοχικών ημερών) |
| leaks | Το αποτέλεσμα της καθημερινής διαφοράς της συνολικής κατανάλωσης ενέργειας από το άθροισμα των τιμών κατανάλωσης ενέργειας που στέλνουν οι αντίστοιχοι αισθητήρες ανά 15 λεπτά και η αντίστοιχη διαφορά για την κατανάλωση νερού |
| totalMovements | Συνολικές ανιχνεύσεις κίνησης μέχρι την δεδομένη ημερομηνία |
| lateRejected | Τα δεδομένα του αισθητήρα νερού w1 που ήρθαν καθυστερημένα κατά 10 μέρες |

Fig4: Τα δεδομένα που στέλνονται σε κάθε topic

Για την αναγνώριση των late rejected events χρησιμοποιήσαμε έναν transformer ο οποίος αποθηκεύει την μέγιστη ημερομηνία που έχει συναντήσει μέχρι στιγμής και αν η ημερομηνία του νέου δεδομένου είναι τουλάχιστον 10 μέρες παλαιότερη αναγνωρίζεται ως late rejected και φιλτράρεται εκτός από την επεξεργασία των δεδομένων και αποστέλλεται σε ξεχωριστό topic στον message broker.

Αντίστοιχα δημιουργήθηκε ένας transformer που αποθηκεύει πόσες ανιχνεύσεις κίνησης έχουν γίνει μέχρι εκείνη την στιγμή και προσθέτει 1 για κάθε νέα ανίχνευση που λαμβάνει.

Tέλος για τον υπολογισμό των διαρροών γίνεται inner join των stream που υπολογίζουν το sum με τα αντίστοιχα stream που περιέχουν τις καθημερινές συνολικές μετρήσεις κατανάλωσης ενέργειας και νερού αντίστοιχα.

1. Το component influxdb αποτελεί την timeseries βάση δεδομένων στην οποία αποθηκεύονται τόσο το αρχικά όσο και τα επεξεργασμένα δεδομένα.

Το component telegraf λαμβάνει τα δεδομένα από τα κατάλληλα topic σε μορφή json και κάνει τις απαραίτητες αλλαγές τα στέλνει σε μορφή influx τόσο στην βάση για την εισαγωγή τους, όσο και μέσω web sockets στο Grafana για την απεικόνιση τους.

Μία από τις βασικές αλλαγές που κάνει το telegraf είναι να παίρνει την ημερομηνία της κάθε εγγραφής που λαμβάνει από τα field σε millisecond και να την μεταφέρει στο πεδίο timestamp, προκειμένου να αποθηκευτεί η εγγραφή με την σωστή ημερομηνία στην βάση δεδομένων και όχι με την ημερομηνία εισαγωγής της σε αυτή.

Ακόμη στα δεδομένα που είναι διαθέσιμο το όνομα του αισθητήρα στον οποίο αναφέρονται, μεταφέρεται στο όνομα του measurement μαζί με το αντίστοιχο prefix που αναφέρεται στον τύπο των δεδομένων, ώστε να είναι ευδιάκριτα τα ονόματα στην βάση δεδομένων και να αντιπροσωπεύουν τα δεδομένα που περιέχουν.

Τέλος στις διαρροές προστέθηκε το είδος την κατανάλωσης στην οποία αναφέρεται, το οποίο λαμβάνεται και πάλι από το αντίστοιχο πεδίο στα fields της κάθε εγγραφής.

1. Το component grafana-server2 είναι υπεύθυνο για την απεικόνιση των δεδομένων σε διαγράμματα και πίνακες. Αρχικά δημιουργούμε ένα API key, με το οποίο θα μπορεί το component telegraf να στέλνει δεδομένα μέσω web sockets για live απεικόνιση. Ακόμη γίνεται σύνδεση της βάσης δεδομένων influxdb, προκειμένου να λαμβάνουμε δεδομένα και από εκεί για απεικόνιση.

Δημιουργούμε ένα dashboard που περιλαμβάνει panels για live απεικονίσεις. Πιο συγκεκριμένα περιλαμβάνει time series απεικονίσεις για τους αισθητήρες TH1, W1, Wtot, τα αποτελέσματα των aggregation για TH1, W1, την διαρροή του νερού, αλλά και πίνακες για τον TH1, την διαρροή νερού και τα Late rejected events.

Ακόμη δημιουργούμε ένα dashboard που περιλαμβάνει αντίστοιχα panels, όπως στην live παρουσίαση, για απεικονίσεις δεδομένων από την βάση.

Στο τέλος κάνουμε mount το image του component, ώστε να μην χρειάζεται να τα δημιουργούμε όλα ξανά από την αρχή κάθε φορά, αλλά να μπορούμε να έχουμε πρόσβαση με εύκολο τρόπο απλώς πατώντας στα link που περιλαμβάνονται στα βήματα για να σετάρουμε την εφαρμογή μας.

Παρακάτω στις εικόνες 5 και 6 παρουσιάζονται τόσο τo component diagram, το οποίο περιλαμβάνει όλα τα components αλλά και το πως συνδέονται μεταξύ τους, όσο και το deployment diagram το οποίο παρουσιάζει την δομή του συστήματος, αλλά και τα components που υπάρχουν σε κάθε κόμβο[2].

Το software που χρησιμοποιήθηκε

* Python3.10.10 για την παραγωγή των δεδομένων των αισθητήρων. Η Python είναι διερμηνευόμενη, γενικού σκοπού και υψηλού επιπέδου, γλώσσα προγραμματισμού και 3.10.10 είναι η τελευταία της έκδοση την δεδομένη στιγμή[3].
* Kafka για την μεταφορά δεδομένων ανάμεσα στα services και την επεξεργασία τους μέσω streams. Kafka είναι μια distributed event streaming πλατφόρμα που χρησιμοποιείται για επεξεργασία ροών δεδομένων[4],[5]. Πιο συγκεκριμένα για την επεξεργασία των δεδομένων γίνεται χρήση των Kafka streams τα οποία αποτελούν μια βιβλιοθήκη ανάπτυξης εφαρμογών, όπου η είσοδος και η

Diagram

Description automatically generatedDiagram

Description automatically generatedFig5: Component diagram της εφαρμογής μας

Fig6: Deployment diagram της εφαρμογής μας

έξοδος αποθηκεύονται σε clusters του kafka[6].

* telegraf που χρησιμοποιήθηκε για την λήψη των δεδομένων από τον kafka και την αποστολή τους στην κατάλληλη μορφή, τόσο για την είσοδο τους στην timeseries βάση δεδομένων, όσο και στην αποστολή τους μέσω websockets στο grafana για την live απεικόνιση τους.

To telegraf είναι ένας ανοιχτού-κώδικα server agent βασίζεται σε plugins και χρησιμοποιείται για την συλλογή και μεταφορά metrics[7].

* influxdb για την αποθήκευση των metrics που στέλνει το telegraf.

InfluxDB είναι μια Τime Series πλατφόρμα δεδομένων που χρησιμοποιείται κυρίως σε εφαρμογές Internet of Things, analytics και cloud[8].

* Grafana για την απεικόνιση των δεδομένων τόσο live όταν προέρχονται μέσω websockets από το telegraf, όσο και χωρίς αυτόματη ενημέρωση όταν προέρχονται από την influxdb

Grafana είναι ένα εργαλείο ανοιχτού-κώδικα συμβατό με διαφορετικούς τύπους υπολογιστών και λειτουργικών συστημάτων και χρησιμοποιείται για analytics και για οπτικοποίηση των δεδομένων. Παρέχει πολλών ειδών διαγράμματα, γραφικές παραστάσεις και ειδοποιήσεις για το web, όταν συνδέεται με υποστηριζόμενες πηγές δεδομένων[9].

# RESULTS-CONCLUSION

Αποτελεσματα στο Grafana, influxDB κτλ

# Acknowledgment

Θα θέλαμε να ευχαριστήσουμε την καθηγήτρια Vassiliki(Verena) Κantere και τον Παρασκευά Κερασιώτη για την πολύτιμη και άμεση βοήθεια τους, καθ’όλη την διάρκεια της εργασίας.

##### References

1. https://el.wikipedia.org/wiki/Διαδίκτυο\_των\_πραγμάτων
2. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-deployment-diagram/
3. https://el.wikipedia.org/wiki/Python
4. https://el.wikipedia.org/wiki/Apache\_Kafka
5. https://kafka.apache.org/
6. https://kafka.apache.org/documentation/streams/
7. https://www.influxdata.com/time-series-platform/telegraf/
8. https://www.influxdata.com/
9. https://en.wikipedia.org/wiki/Grafana