ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ ΤΜΗΜΑΤΟΣ

ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

ΚΑΤΕΥΘΥΝΣΗ ΔΙΑΧΕΙΡΙΣΗ ΔΙΚΤΥΩΝ ΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΥΠΗΡΕΣΙΩΝ

ΕΠΟΜΕΝΗΣ ΓΕΝΙΑΣ

Σύστημα Ειδοποίησης, Παρακολούθησης και Διαχείρισης Θερμοκρασίας και Υγρασίας για ένα Data Center

Μεταπτυχιακή Εργασία

Θεόδωρος Δίπλας

Αθήνα, 2022

HAROKOPIO UNIVERSITY

SCHOOL OF DIGITAL TECHNOLOGY

DEPARTMENT INFORMATICS AND TELEMATICS

POSTGRADUATE PROGRAMME INFORMATICS AND TELEMATICS

COURSE TELECOMMUNICATION NETWORKS AND TELEMATIC SERVICES

Notification, Monitoring and Management System for Datacenter Temperature and Humidity

Master Thesis

Theodoros Diplas

Athens, 2022

ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ ΤΜΗΜΑΤΟΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

ΚΑΤΕΥΘΥΝΣΗ ΔΙΑΧΕΙΡΙΣΗ ΔΙΚΤΥΩΝ ΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΥΠΗΡΕΣΙΩΝ ΕΠΟΜΕΝΗΣ ΓΕΝΙΑΣ

Τριμελής Εξεταστική Επιτροπή

Ανάργυρος Τσαδήμας (Επιβλέπων)

Ε.ΔΙ.Π., ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ, ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

Αλέξανδρος Δημόπουλος (Επιβλέπων)

Λέκτορας, ΣΧΟΛΗ ΝΑΥΤΙΚΩΝ ΔΟΚΙΜΩΝ

Θωμάς Καμαλάκης

Επίκουρος Καθηγητής, ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ, ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

Ο Θεόδωρος Δίπλας

δηλώνω υπεύθυνα ότι:

1) Είμαι ο κάτοχος των πνευματικών δικαιωμάτων της πρωτότυπης αυτής εργασίας και από όσο γνωρίζω η εργασία μου δε συκοφαντεί πρόσωπα, ούτε προσβάλει τα πνευματικά δικαιώματα τρίτων.

2) Αποδέχομαι ότι η ΒΚΠ μπορεί, χωρίς να αλλάξει το περιεχόμενο της εργασίας μου, να τη διαθέσει σε ηλεκτρονική μορφή μέσα από τη ψηφιακή Βιβλιοθήκη της, να την αντιγράψει σε οποιοδήποτε μέσο ή/και σε οποιοδήποτε μορφότυπο καθώς και να κρατά περισσότερα από ένα αντίγραφα για λόγους συντήρησης και ασφάλειας.

3) Όπου υφίστανται δικαιώματα άλλων δημιουργών έχουν διασφαλιστεί όλες οι αναγκαίες άδειες χρήσης ενώ το αντίστοιχο υλικό είναι ευδιάκριτο στην υποβληθείσα εργασία.

ΕΥΧΑΡΙΣΤΙΕΣ

Για τη διεκπεραίωση της παρούσας μεταπτυχιακής εργασίας, θα ήθελα να ευχαριστήσω τους επιβλέποντες καθηγητές **κ. Αλέξανδρο Δημόπουλο** και **κ. Ανάργυρο Τσαδήμα** για την υποστήριξη και καθοδήγησή τους καθ' όλη τη διάρκεια εκπόνησης. Τους ευχαριστώ θερμά επίσης για το ζήλο που επέδειξαν στο διδακτικό τους έργο κατά τη διάρκεια των μαθημάτων, με αποτέλεσμα να μου προσφέρουν τα απαραίτητα εφόδια για να ανταπεξέλθω στις απαιτήσεις αυτής της εργασίας αλλά και στην προσωπική μου εξέλιξη στο τομέα της πληροφορικής.

Οφείλω επίσης να ευχαριστήσω τον συνάδελφο και φίλο πλέον, **κ. Γιώργο Καρούμπαλη** για τα ερεθίσματα και τις συμβουλές που μου προσέφερε απλόχερα στη παρούσα προσπάθεια. Οι εξαιρετικά χρήσιμες συζητήσεις πάνω σε μικροελεγκτές και αισθητήρες αλλά και στον σχεδιασμό του συστήματος με βοήθησαν να λύσω πολλούς από τους προβληματισμούς μου.

Τέλος θα ήθελα να ευχαριστήσω τη σύζυγό μου **Βενετία** και τα παιδιά μου **Κωνσταντίνο** και **Ζωή** για την ανοχή και την στήριξη που μου έδειξαν αφού έπρεπε να αφιερώσω ατελείωτες ώρες στον υπολογιστή προκειμένου να ολοκληρώσω την συγγραφή με τον καλύτερο δυνατό τρόπο.

Σας ευχαριστώ θερμά όλους.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Περίληψη

Η παρούσα μεταπτυχιακή εργασία ασχολείται με την ανάπτυξη και υλοποίηση ενός συστήματος ειδοποίησης, παρακολούθησης και διαχείρισης θερμοκρασίας και υγρασίας ενός Data Center. Ο κύριος σκοπός της είναι η δημιουργία και περιγραφή ενός αυτόνομου συστήματος, που θα έχει την δυνατότητα να αποθηκεύει και να οπτικοποιεί ενδείξεις που λαμβάνει από αισθητήρες που βρίσκονται εντός του Data Center. Επιπροσθέτως το σύστημα θα μπορεί να ενεργεί αν οι ενδείξεις από τους αισθητήρες ξεπεράσουν κάποιο προκαθορισμένο κατώφλι.

Η υλοποίηση περιλαμβάνει ένα Raspberry Pi που θα αναλάβει την υποστήριξη του απαραίτητου λογισμικού που απαιτείται για την αποθήκευση, επεξεργασία και οπτικοποίηση των δεδομένων που θα λαμβάνει από τους αισθητήρες. Για την ασύρματη διασύνδεση των αισθητήρων χρησιμοποιήθηκαν μικροελεγκτές ESP8266 που προγραμματίστηκαν κατάλληλα χρησιμοποιώντας το περιβάλλον ανάπτυξης του Arduino IDE. Οι αισθητήρες που χρησιμοποιήθηκαν στην παρούσα φάση της υλοποίησης είναι θερμοκρασίας, υγρασίας, νερού, κίνησης και μαγνητικοί. Στην αγορά διατίθεται πλήθος αισθητήρων που μπορούν να κατηγοριοποιηθούν σύμφωνα με το είδος, το κόστος και την ακρίβεια των μετρήσεων που πετυχαίνουν.

Η ανάπτυξη του εν λόγω συστήματος έγινε με γνώμονα την όσο δυνατόν πιο εύκολη εγκατάσταση, παραμετροποίηση αλλά και επέκτασή του με περισσότερους αισθητήρες. Για τον λόγο αυτό χρησιμοποιήθηκαν εργαλεία και τεχνολογίες όπως η ansible, το docker, το πρωτόκολλο MQTT και το node-red.

Επιπλέον στον σύνδεσμο GitHub (https://github.com/theohitman/pms-thesis) υπάρχουν οδηγίες και κώδικας για την υλοποίηση σεναρίου χρήσης.

Λέξεις κλειδιά: Raspberry Pi, ESP8266, Sensors, Containers, Ansible

Abstract

This master thesis deals with the development and implementation of a system for alerting, monitoring and managing the temperature and humidity of a Data Center. Its main purpose is to create and describe an autonomous system, which has the ability to store and visualize signals received from sensors located within the Data Center. In addition, the system will be able to intervene in case indications from the sensors exceed a predetermined threshold.

The implementation includes a Raspberry Pi supporting the necessary software used to store, process and visualize the data received from the sensors. ESP8266 microcontrollers participated to wirelessly connect the sensors and programmed using the Arduino IDE environment. The sensors used in the present phase of the implementation are temperature, humidity, water, motion and magnetic. There are a great number of sensors available on the market that can be categorized according to the type, cost and accuracy of the measurements they achieve.

The development of this system was based on the easiest installation, configuration and expansion ability with more sensors. To achieve this goal, tools and technologies such as ansible, docker, MQTT and node-red participated.

Following GitHub (https://github.com/theohitman/pms-thesis) instructions and code are available for the implementation of a use case scenario.

Keywords: Raspberry Pi, ESP8266, Sensors, Containers, Ansible

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ/ΑΚΡΩΝΥΜΙΑ

ARM	Advanced RISC Machine
CLI	Command Line Interface
DHCP	Dynamic Host Configuration Protocol
DIY	Do It Yourself
DNS	Domain Name System
GPIO	General Purpose Inputs Outputs
HTTP	Hyper Text Transfer Protocol
IoT	Internet of Things
IP	Internet Protocol
MAC	Media Access Control
MQTT	Message Queuing Telemetry Transport
PUE	Power Usage Effectiveness
SBC	Single Board Computer
SMS	Short Message Service
SoC	System on a Chip
SSH	Secure Shell
TCP	Transmission Control Protocol
TLS	Transport Layer Security
USB	Universal Serial Bus

ΕΙΣΑΓΩΓΗ

Τα κέντρα δεδομένων (Data Centers) είναι εγκαταστάσεις που συγκεντρώνουν τον εξοπλισμό ενός οργανισμού με σκοπό την αποθήκευση, επεξεργασία και διάδοση πληροφοριών. Είναι ζωτικής σημασίας για την βιωσιμότητα ενός οργανισμού διότι στεγάζουν εξοπλισμό και δεδομένα που είναι απαραίτητα για τις καθημερινές λειτουργίες του. Κατά συνέπεια, η ασφάλεια και η αξιοπιστία των Data Centers είναι μεταξύ των κορυφαίων προτεραιοτήτων κάθε οργανισμού. Για την απρόσκοπτη λειτουργία τους απαιτείται η ύπαρξη ενός αξιόπιστου συστήματος ελέγχου και παρακολούθησης των επικρατούντων περιβαλλοντολογικών συνθηκών εντός της εγκατάστασης. Τα τελευταία χρόνια έχει γίνει μεγάλη προσπάθεια για βελτίωση της ενεργειακής απόδοσης (Power Usage Effectiveness - PUE) των Data Centers ενώ κλειδί σε αυτή την προσπάθεια είναι η ύπαρξη ιδανικών συνθηκών θερμοκρασίας και υγρασίας στο εσωτερικό τους [1].

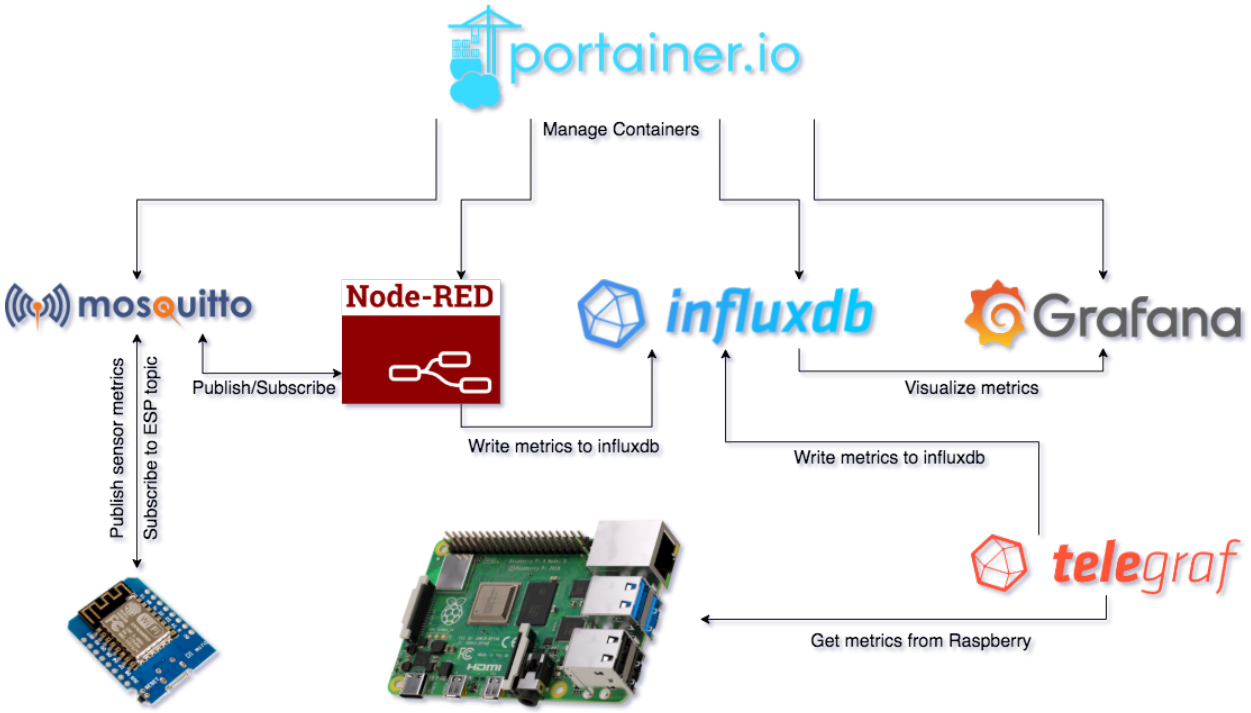
Με γνώμονα τα παραπάνω και στο πλαίσιο της παρούσας μεταπτυχιακής εργασίας θα γίνει προσπάθεια ανάπτυξης ενός αυτόνομου συστήματος παρακολούθησης περιβαλλοντολογικών συνθηκών σε ένα κέντρο δεδομένων με όσο το δυνατόν απλούστερο υλικό και λογισμικό ανοιχτού κώδικα.

Στο πρώτο κεφάλαιο της εργασίας θα γίνει η περιγραφή της αρχιτεκτονικής του συστήματος ενώ στο δεύτερο κεφάλαιο θα συνεχιστεί η περιγραφή με τις τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν. Στο τρίτο κεφάλαιο θα γίνει η παραμετροποίηση του συστήματος ενώ στο τέταρτο κεφάλαιο της εργασίας θα αναπτυχθεί ένα σενάριο χρήσης με αισθητήρες θερμοκρασίας, υγρασίας, νερού, κινήσεως και μαγνητικούς για ένα μικρό data center. Τέλος θα αναφερθούν οι δυσκολίες που προέκυψαν κατά την υλοποίηση της εργασίας, μελλοντικές βελτιώσεις που θα μπορούσαν να εφαρμοστούν καθώς και τα τελικά συμπεράσματα.

ΚΕΦ.1: ΠΕΡΙΓΡΑΦΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΚΑΙ ΥΛΙΚΟΥ

1.1. Γενική Αρχιτεκτονική

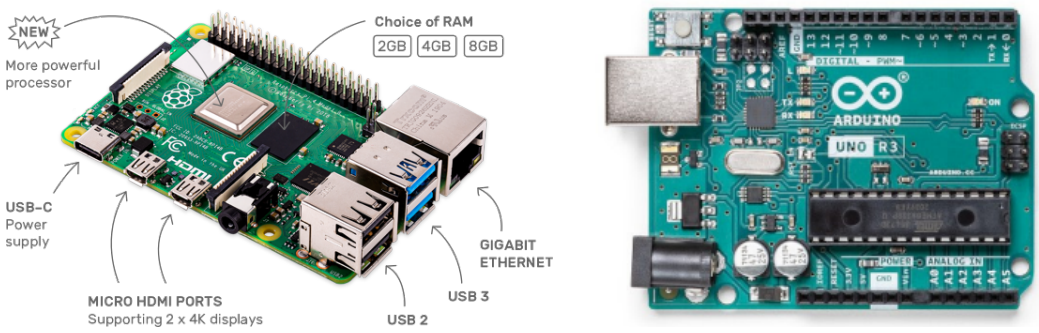
Το σύστημα αποτελείται από ένα Raspberry Pi με εγκατεστημένο docker και docker compose για την υποστήριξη των επιμέρους εφαρμογών που εκτελούνται σε containers. Ένας telegraf agent που είναι και αυτός εγκατεστημένος στο raspberry παίρνει μετρήσεις που αφορούν την υγεία του raspberry και τις στέλνει στην influxdb που αποτελεί την βάση δεδομένων του συστήματος για αποθήκευση και μελλοντική αναφορά. Για την υποστήριξη της επικοινωνίας με το πρωτόκολλο MQTT χρησιμοποιείται το mosquito που είναι ένας message broker. Οι διάφοροι αισθητήρες είναι συνδεδεμένοι ενσύρματα με πλακέτες ESP8266 όπου είναι σε θέση να στέλνουν ασύρματα τις ενδείξεις που λαμβάνουν με MQTT μηνύματα στην influxdb. Το Node-RED αναλαμβάνει την ενορχήστρωση όλης της λειτουργικότητας προσφέροντας ταυτόχρονα εύκολη παραμετροποίηση. Το Grafana έχει αναλάβει τον ρόλο της οπτικοποίησης των δεδομένων αφού τα διαβάσει από την influxdb. Τέλος το portainer χρησιμοποιείται για τη διαχείριση και παρακολούθηση όλων των containers. Στο κεφάλαιο 2 γίνεται αναλυτική περιγραφή όλων των τεχνολογιών που αναφέρθηκαν και χρησιμοποιήθηκαν.



Σχ. 1. Αρχιτεκτονική Συστήματος

1.2. Raspberry Pi

Δύο από τις πιο δημοφιλείς πλατφόρμες ανάπτυξης συστημάτων για IoT εφαρμογές είναι το Raspberry Pi και το Arduino [2] [3] . Λιγότερο διαδεδομένες είναι η Intel Edison, η Particle Photon η Tessel, η Adafruit Feather M0 και πολλές ακόμα. Η πλατφόρμα Arduino μάλιστα διατίθεται σε πολλές διαφορετικές παραλλαγές ανάλογα με το είδος και το πλήθος των συσκευών που θα χρειαστεί να υποστηριχθούν στο εκάστοτε project. Μια ακόμα πλατφόρμα τύπου Arduino που το τελευταίο διάστημα χρησιμοποιείται κατά κόρον είναι το ESP8266 και ο διάδοχός του το ESP32. Είναι σύνηθες στα συστήματα IoT εφαρμογών να απαιτείται η χρήση διαφορετικών τύπων πλατφόρμας. Στην παρούσα εργασία θα χρησιμοποιηθεί το Raspberry Pi που θα παίζει τον ρόλο του server τρέχοντας τις απαραίτητες εφαρμογές και το ESP8266 για την ασύρματη διασύνδεση και υποστήριξη των αισθητήρων.



Εικ. 1. Raspberry Pi vs Arduino UNO

Ένα Arduino δεν θα μπορούσε να υποστηρίξει τις εφαρμογές που χρειάζονται για την υλοποίηση της εργασίας καθώς πρόκειται απλώς για μια πλακέτα με ενσωματωμένο μικροελεγκτή και εισόδους/εξόδους χωρίς λειτουργικό σύστημα. Σκοπός της είναι να υποστηρίζει συσκευές εκτελώντας επαναλαμβανόμενα έναν συγκεκριμένο κώδικα, σαν ένα τυπικό ενσωματωμένο σύστημα.

1.3. IoT Development Boards

Τα development boards ή πλακέτες ανάπτυξης είναι τυπωμένα κυκλώματα που διασυνδέουν υλικό για να υποστηρίξουν μικροελεγκτές. Αποτελούνται από κύκλωμα τροφοδοσίας ρεύματος, διεπαφή για προγραμματισμό, βασικές εισόδους-εξόδους (κουμπιά, led) και GPIO pins για διασύνδεση αισθητήρων, μοτέρ, οθονών κτλ. [4]. Τα βασικά χαρακτηριστικά που κάνουν ένα development board να ξεχωρίζει είναι τα εξής:

- Δυνατότητες διασύνδεσης (Wi-Fi, Bluetooth, Ethernet, etc.)
- Επεκτασιμότητα
- Επεξεργαστική Ισχύς
- Μνήμη
- Υποστήριξη περιφερειακών
- Κόστος
- Μέγεθος

Τα development boards θα μπορούσαν να χωριστούν σε 3 κατηγορίες:

1. Πλακέτες μικροελεγκτών που χρησιμοποιούνται κυρίως σε εμφυτεύσιμες ιατρικές συσκευές, μηχανές γραφείου, ηλεκτρικά εργαλεία κτλ.
2. System-on-Chip (SoC) που περιέχουν όλα τα βασικά ηλεκτρονικά εξαρτήματα και κυκλώματα ενός συστήματος και βρίσκουν εφαρμογή σε κινητές συσκευές, δικτυακές συσκευές και IoT.
3. Single-board computers (SBC) που έχουν όλα τα απαραίτητα συστατικά ενός υπολογιστή όπως μνήμη, επεξεργαστή, συσκευές εισόδου και εξόδου πάνω σε μια πλακέτα [5].

1.3.1. ESP8266

Το ESP8266 είναι ένα SOC με ενσωματωμένο Wi-Fi και υποστήριξη της στοίβας πρωτοκόλλου TCP/IP [6]. Το μικρό του μέγεθος, το χαμηλό κόστος και η δυνατότητα να παρέχει ασύρματη επικοινωνία το έχει κάνει ιδιαίτερα δημοφιλές για IoT λύσεις. Χρησιμοποιείται ευρέως σε έξυπνες συσκευές ασφαλείας συμπεριλαμβανομένων των καμερών παρακολούθησης και των έξυπνων κλειδαριών, σε έξυπνες βιομηχανικές συσκευές (PLCs), έξυπνες ενεργειακές και ιατρικές συσκευές κτλ. [7]. Υποστηρίζονται πολλές διαφορετικές γλώσσες προγραμματισμού για να προγραμματίσει κάποιος το ESP8266 μεταξύ των οποίων είναι η C, η LUA και η Micro Python αλλά συνήθως χρησιμοποιείται το Arduino IDE με χρήση της γλώσσας C++ που περιγράφεται παρακάτω.

Εικ. 2. WeMos D1 mini with ESP8266

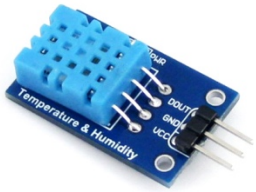
Στην παρούσα εργασία επιλέχθηκε το Wemos D1 mini ESP8266 που είναι μια πλακέτα προγραμματισμού μέρος της οποία είναι το ESP8266. Η πλακέτα αυτή δίνει την δυνατότητα σύνδεσης με microUSB για τον προγραμματισμό αλλά και την τροφοδοσία της με ρεύμα. Σχεδιάστηκε για να είναι όσο το δυνατόν μικρότερη σε μέγεθος ενώ προσφέρει 11 ψηφιακά pins εισόδου/εξόδου και 1 αναλογικό pin εισόδου για διασύνδεση με αισθητήρες και άλλες συσκευές. Ο χρονισμός του ρολογιού είναι στα 80MHz και η μνήμη flash στα 4MB που είναι αρκετά για να υποστηρίξουν τους αισθητήρες που απαιτούνται και να στέλνει ασύρματα τις ενδείξεις των αισθητήρων στο raspberry. Ο προγραμματισμός της γίνεται απευθείας από το Arduino IDE αλλά και με το Visual Studio Code χρησιμοποιώντας το extension PlatformIO.

Άλλες παρόμοιες πλακέτες είναι η ESP8266 12-E NodeMCU ενώ ακόμα οικονομικότερη είναι η ESP8266 ESP-01 η οποία όμως δεν συστήνεται διότι έχει μόνο 4 ψηφιακά pins εισόδου/εξόδου γεγονός που περιορίζει το πλήθος των αισθητήρων που μπορεί να υποδεχτεί όπως επίσης και δεν διαθέτει θύρα USB οπότε και απαιτείται μετατροπέας από USB σε σειριακό για τον προγραμματισμό της.

1.4. Αισθητήρες

1.4.1. Temperature-Humidity Sensor DHT11

Για τη μέτρηση της θερμοκρασίας και της υγρασίας χρησιμοποιήθηκε ο αισθητήρας DHT11. Πρόκειται για έναν αργό αλλά οικονομικό αισθητήρα με απόκλιση ακρίβειας της θερμοκρασίας έως 2 βαθμούς Κελσίου. Αντιλαμβάνεται την υγρασία μετρώντας την ηλεκτρική αντίσταση ανάμεσα σε 2 ηλεκτρόδια ενώ την θερμοκρασία με μια thermistor αντίσταση που είναι ευαίσθητη στην αλλαγή θερμοκρασίας.



Εικ. 3. DHT11 Sensor

Διαθέτει τρεις ακροδέκτες. Ο πρώτος συνδέεται με τη παροχή τάσης, ο δεύτερος στη γείωση ενώ ο τρίτος συνδέεται με έναν ψηφιακό ακροδέκτη στην πλακέτα ESP8266 όπου θα πηγαίνουν οι μετρήσεις. Τα βασικά τεχνικά χαρακτηριστικά του αισθητήρα φαίνονται στον Πίνακα 1 ενώ για πιο γρήγορες και εγκυρότερες μετρήσεις συστήνεται ο αισθητήρας DHT22 ή BME280.

Τάση Τροφοδοσίας	3.3V ή 5.5V
Ρεύμα Λειτουργίας	0.3mA
Εύρος μέτρησης τιμών θερμοκρασίας:	0 °C έως 50 °C
Εύρος μέτρησης τιμών υγρασίας:	20%H έως 90%H

Πίν. 1. Τεχνικά χαρακτηριστικά αισθητήρα DHT11

1.4.2. PIR Sensor Module HC-SR501

Εικ. 4. HC-SR501 Sensor

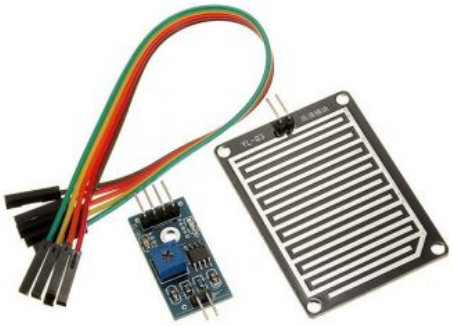
Για την ανίχνευση δραστηριότητας εντός του data center χρησιμοποιήθηκε ένας αισθητήρας ανίχνευσης κίνησης HC-SR501. Πρόκειται για έναν παθητικό ανιχνευτή υπέρυθρης ακτινοβολίας που είναι σε θέση να εντοπίζει αλλαγές στη θερμοκρασία που μπορούν να προκληθούν από την παρουσία ατόμου, ζώου ή αντικειμένου στο χώρο. Διαθέτει και αυτός τρεις ακροδέκτες για τάση, γείωση και ένδειξη HIGH ή LOW αν εντοπίσει κίνηση εντός του εύρους ανίχνευσής του.

Τάση Τροφοδοσίας	5V
Ρεύμα Λειτουργίας	60mA
Εύρος ανίχνευσης	7 μέτρα με κώνο 120 μοιρών
Ευαισθησία και Χρονοκαθυστέρηση	Προσαρμόζονται

Πίν. 2. Τεχνικά χαρακτηριστικά αισθητήρα HC-SR501

Οι αισθητήρες αυτού του τύπου απαιτούν 30 - 60 δευτερόλεπτα για να εγκλιματιστούν στον χώρο ενώ ανάμεσα στις μετρήσεις υπάρχει μια κενή περίοδος 5 με 6 δευτερόλεπτα στην οποία δεν μπορούν να εντοπίσουν κίνηση. Ο αισθητήρας αυτός χρησιμοποιήθηκε στην εργασία περισσότερο για λόγους ασφαλείας ώστε να υπάρχει κάποια ειδοποίηση αν εισέλθουν στο χώρο άτομα σε μη επιτρεπόμενες ώρες.

1.4.3. Rain Sensor Module



Εικ. 5. Rain Sensor

Αποτελείται από 2 πλακέτες που είναι συνδεδεμένες μεταξύ τους. Η μεγαλύτερη πλακέτα λειτουργεί σαν διακόπτης και είναι αυτή που ανιχνεύει την ύπαρξη νερού ενώ η μικρότερη πλακέτα λειτουργεί σαν μονάδα ελέγχου μετατρέποντας τις αναλογικές ενδείξεις σε ψηφιακές. Όταν πέσει μια σταγόνα νερού στον αισθητήρα κλείνει το κύκλωμα επιτρέποντας το ρεύμα να περάσει. Τότε δίνεται σήμα HIGH στην ψηφιακή έξοδο ενώ στην αναλογική δίνεται τάση ανάλογα με την ποσότητα νερού που εκτίθεται στον αισθητήρα. Εκτός από τον ψηφιακό και αναλογικό ακροδέκτη υπάρχουν άλλοι δύο ακροδέκτες που απαιτούν τάση και γείωση για να λειτουργήσει σωστά ο αισθητήρας.

Τάση Τροφοδοσίας	5V
Ρεύμα Λειτουργίας	100mA max
Επιφάνεια Αισθητήρα	55x40mm
Ευαισθησία	Προσαρμόζεται

Πίν. 3. Τεχνικά χαρακτηριστικά αισθητήρα Rain Sensor

Η παρουσία νερού σε ένα data center μπορεί να είναι καταστροφική. Το νερό μπορεί να διεισδύσει κάτω από το υπερυψωμένο δάπεδο χωρίς να γίνει αντιληπτό από τους υπευθύνους μέχρι να είναι αργά. Για τον λόγο αυτό η ύπαρξη ενός τέτοιου αισθητήρα κρίνεται απαραίτητη για την εύρυθμη λειτουργία του data center οπότε και προσαρμόστηκε στην εργασία.

1.4.4. Magnetic Reed Switch

Εικ. 6. Magnetic Reed Switch

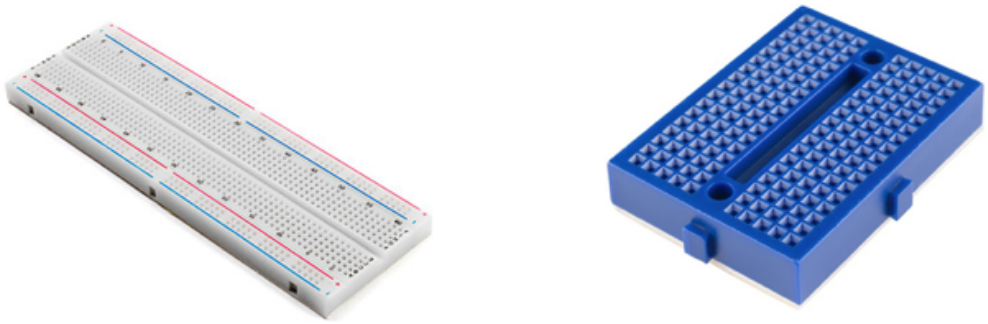
Πρόκειται για έναν μαγνητικό διακόπτη επαφής που χρησιμοποιείται κατά κόρον στα συστήματα ασφαλείας και διασφαλίζει ένδειξη με την κατάσταση μιας πόρτας ή ενός παραθύρου. Στην παρούσα εργασία θα τοποθετηθεί στην κεντρική είσοδο του data center με σκοπό την παρακολούθησή της. Για την σύνδεσή του απαιτείται μία αντίσταση 10kOhms. Η λειτουργία του μοιάζει με αυτή του κοινού διακόπτη με την διαφορά ότι χρησιμοποιεί ένα μαγνητικό πεδίο για το άνοιγμα και κλείσιμο των επαφών του.

Δυνατότητα μεταγωγής	10W
Ρεύμα μεταγωγής	500mA
Διαμόρφωση επαφών	SPST-NO
Τάση μεταγωγής	Max. 110V

Πίν. 4. Τεχνικά χαρακτηριστικά αισθητήρα Reed Switch

1.5 Breadboard

Είναι μια απλή συσκευή που έχει σχεδιαστεί για την δημιουργία κυκλωμάτων χωρίς την απαίτηση κολλήσεων στα επιμέρους ηλεκτρονικά στοιχεία [8]. Υπάρχει σε διάφορα μεγέθη και χρησιμοποιείται ευρέως από αρχάριους αλλά και πιο έμπειρους χρήστες. Για τις ανάγκες τις εργασίας θα χρησιμοποιηθούν breadboards που θα βοηθήσουν στην διασύνδεση των πλακετών ESP8266 με τους απαραίτητους αισθητήρες, led και buzzer.



Εικ. 7. Breadboards

1.6 Sonoff

Είναι ένας έξυπνος διακόπτης τροφοδοσίας ρεύματος που μπορεί να χρησιμοποιηθεί σε μια μεγάλη γκάμα συσκευών. Μια λάμπα, ένα πορτατίφ, ένας θερμοστάτης και ο θερμοσίφωνας είναι μερικές από αυτές. Παρέχει στους χρήστες την δυνατότητα απομακρυσμένου ελέγχου αφού μεταδίδει δεδομένα σε πλατφόρμα cloud μέσω της ασύρματης διασύνδεσή του με WiFi.

Εικ. 8. Sonoff Basic R2

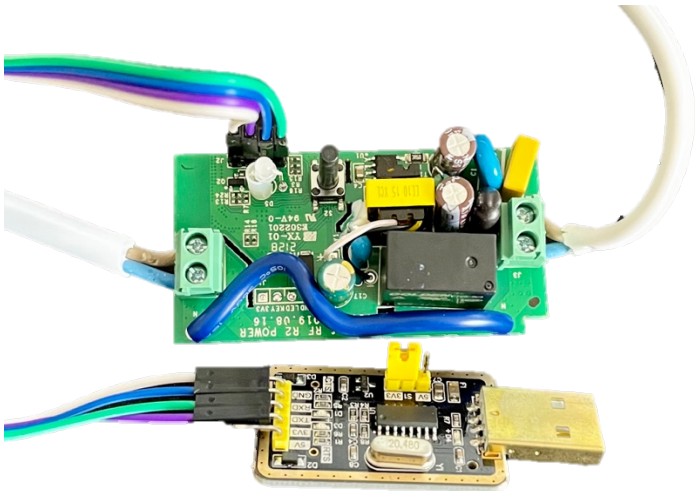
Τάση Τροφοδοσίας	90V - 250V AC
Μέγιστο Ρεύμα Λειτουργίας	10A
Ασύρματα Πρότυπα	Wi-Fi 2.4GHz b/g/n
Μηχανισμοί Ασφάλειας	WEP/WPA-PSK/WPA2-PSK

Πίν. 5. Τεχνικά χαρακτηριστικά Sonoff

Για τις ανάγκες τις εργασίας φορτώθηκε διαφορετικό firmware από το υπάρχον στον διακόπτη Sonoff. Το firmware που επιλέχθηκε είναι το ESP Easy διότι δίνει την δυνατότητα επικοινωνίας και ελέγχου του Sonoff με MQTT μηνύματα καθώς και διασύνδεση πρόσθετων αισθητήρων [9]. Ένας μετατροπέας USB-To-Serial όπως αυτός που φαίνεται στην παρακάτω φωτογραφία είναι απαραίτητος για την διασύνδεση του υπολογιστή με το Sonoff ώστε να γίνει η αντικατάσταση του firmware.

Εικ. 9. USB-To-Serial Adapter

Επιπλέον στο Sonoff θα πρέπει να γίνουν οι κατάλληλες κολλήσεις για την διασύνδεση με τον μετατροπέα. Τέλος θα πρέπει να γίνουν οι κατάλληλες ρυθμίσεις στο Sonoff για σύνδεση με το επιθυμητό ασύρματο δίκτυο, ορισμού της IP διεύθυνσης του MQTT broker, κ.α. [10]. Στη παρακάτω εικόνα φαίνεται το εσωτερικό του διακόπτη και η διασύνδεσή του με τον μετατροπέα με σκοπό τον προγραμματισμό του.



Εικ. 10. Sonoff connected with USB-To-Serial Adapter

ΚΕΦ.2: ΠΕΡΙΓΡΑΦΗ ΤΕΧΝΟΛΟΓΙΩΝ

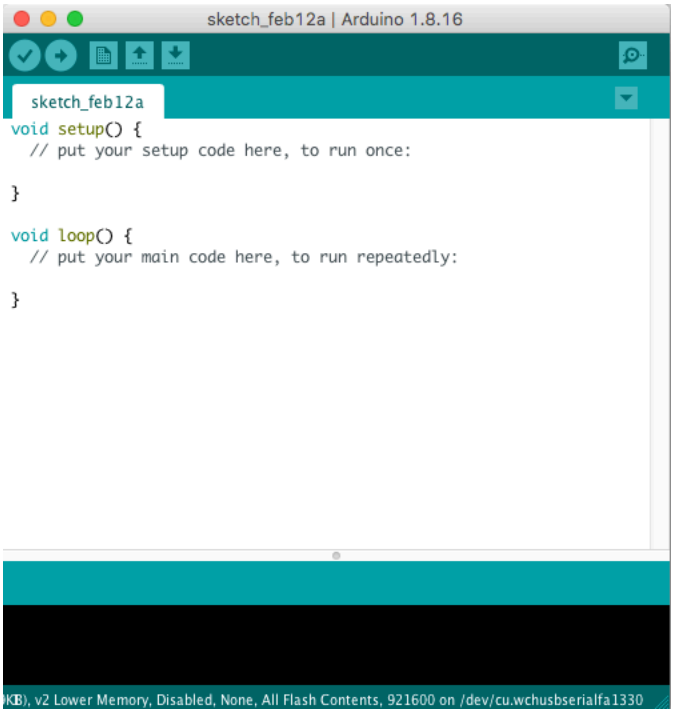
2.1. Raspberry Pi OS

Το Raspberry Pi OS είναι ένα λειτουργικό σύστημα για το Raspberry Pi που βασίζεται στο Debian. Τον Μάιο του 2020 ανακοινώθηκε και 64-bit έκδοση η οποία μέχρι και τον Ιανουάριο του 2022 ήταν σε BETA μορφή [11]. Η Desktop έκδοση του λειτουργικού έχει γραφικό περιβάλλον ενώ η Lite όχι. Επίσης υποστηρίζει docker και docker-compose που είναι τεχνολογίες που θα χρησιμοποιηθούν στην εργασία. Το γεγονός ότι ο επεξεργαστής του είναι ARM αρχιτεκτονικής υποχρεώνει και τα containers που θα χρησιμοποιηθούν να το υποστηρίζουν. Προτιμήθηκε η Lite έκδοση του λειτουργικού διότι απαιτεί λιγότερα resources, ενώ από άποψη ασφάλειας μειώνει την επιφάνεια επίθεσης.

2.2. Arduino IDE

Πρόκειται για ένα ολοκληρωμένο περιβάλλον ανάπτυξης ανοιχτού κώδικα το οποίο χρησιμοποιείται κυρίως για δύο λόγους. Ο πρώτος είναι η συγγραφή και μεταγλώττιση προγραμμάτων και ο δεύτερος είναι η φόρτωση του κώδικα στις πλακέτες. Υποστηρίζει πλακέτες Arduino, συμβατές με Arduino αλλά και πλακέτες από άλλους κατασκευαστές, ενώ μπορεί να εγκατασταθεί σε λειτουργικά συστήματα Windows, Linux και macOS.

Ο κώδικας για Arduino είναι γραμμένος σε C++ με κάποιες προσθήκες μεθόδων και συναρτήσεων [12]. Το Arduino IDE περιέχει ενσωματωμένες βιβλιοθήκες που παρέχουν βασική λειτουργικότητα στις πλακέτες ενώ υπάρχει η δυνατότητα να προστεθούν επιπλέον βιβλιοθήκες που επεκτείνουν τις δυνατότητες των πλακετών. Η γενική δομή ενός προγράμματος για Arduino που λέγεται και sketch έχει ως εξής: Στην αρχή δηλώνονται οι βιβλιοθήκες που θα χρησιμοποιηθούν και στη συνέχεια οι μεταβλητές. Ο βασικός κορμός του κώδικα αποτελείται από δύο υποχρεωτικές συναρτήσεις. Η πρώτη είναι η setup() η οποία τρέχει μια φορά κατά την εκκίνηση ή επανεκκίνηση της πλακέτας και η δεύτερη είναι η loop() η οποία επαναλαμβάνεται καθ’ όλη τη διάρκεια λειτουργίας της.



Εικ. 11. Arduino IDE

Το βασικό παράθυρο του Arduino IDE όπως φαίνεται και στην Εικόνα 7 θα χρησιμοποιηθεί για να φορτωθεί ο απαιτούμενος κώδικας στο Wemos D1 mini ESP8266.

2.3. Ansible

Η ansible είναι ένα εργαλείο ανοιχτού κώδικα που επιτρέπει απομακρυσμένο έλεγχο και παραμετροποίηση πολλών servers ταυτόχρονα από ένα σημείο. Αυτοματοποιεί διαδικασίες όπως την εγκατάσταση μιας εφαρμογής μέσα από αρχεία yaml που ονομάζονται playbooks. Τα βήματα μιας διαδικασίας ορίζονται μια φορά μέσα στο playbook ενώ το playbook μπορεί να επαναχρησιμοποιηθεί όσες φορές και για όσους servers απαιτηθεί. Με αυτόν τον τρόπο αποφεύγονται ανθρώπινα λάθη και τυχόν παραλείψεις κατά την επανάληψη. Βασίζεται σε modules που είναι μικρά προγράμματα και χρησιμοποιούνται ανάλογα με την εργασία που πρέπει να γίνει. Ένα ακόμα δυνατό σημείο της ansible είναι ότι όσες φορές και να τρέξει ένα playbook η παραμετροποίηση στους servers θα γίνει μόνο όπου χρειάζεται ενώ το τελικό αποτέλεσμα θα παραμείνει το ίδιο.

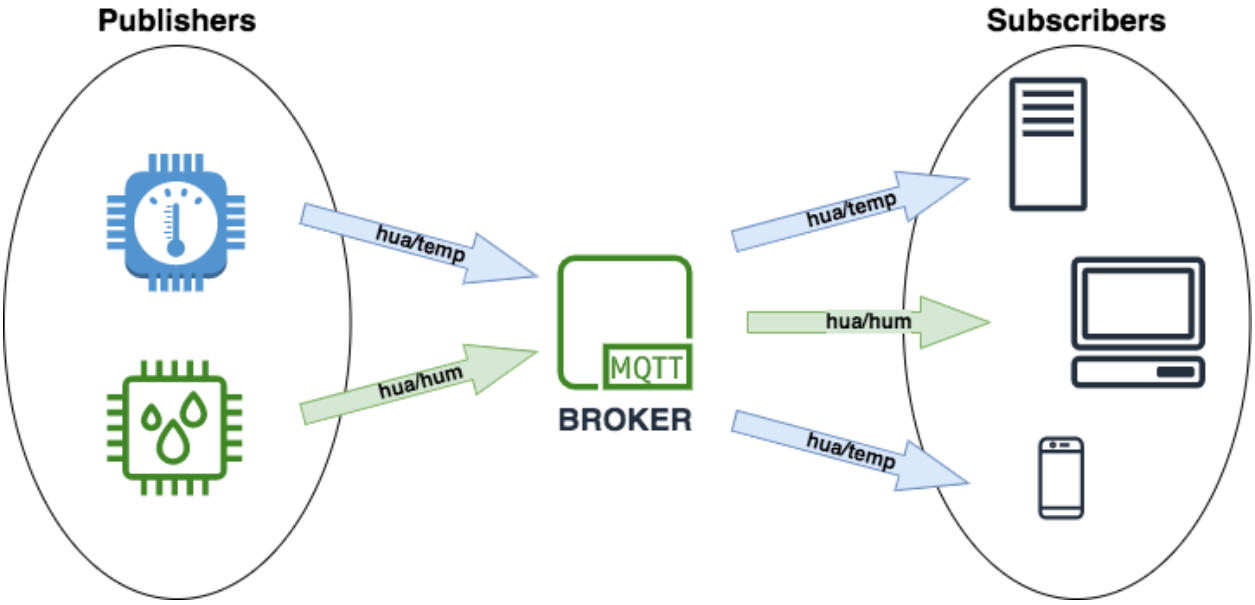
Χρησιμοποιήθηκε στην εργασία για την αυτοματοποίηση των επιμέρους διαδικασιών και την ευκολότερη εγκατάσταση του συστήματος.

2.4. Mosquitto

Είναι ένα λογισμικό ανοιχτού κώδικα που υλοποιεί έναν μεσίτη μηνυμάτων (message broker) και κάνει χρήση του πρωτοκόλλου MQTT. Έχει χαμηλές απαιτήσεις σε υλικό και είναι ιδανικό για χρήση σε SBC συσκευές. Θα αποτελέσει τον βασικό πυλώνα επικοινωνίας μεταξύ των αισθητήρων και του raspberry.

2.4.1. MQTT Protocol

Το πρωτόκολλο MQTT προσφέρει μια μέθοδο ανταλλαγής μηνυμάτων με το μοντέλο δημοσίευσης/εγγραφής (publish/subscribe) που περιγράφεται παρακάτω. Χρησιμοποιείται ευρέως στις IoT συσκευές λόγω του ελαφρύ του σχεδιασμού και του ελάχιστου εύρους ζώνης δικτύου που απαιτεί [13].



Σχ. 2. MQTT Publish/Subscribe Model

Είναι ένα πρωτόκολλο αμφίδρομης επικοινωνίας που αποτελείται από τρεις βασικές οντότητες:

- Τους **publishers** που είναι συνήθως IoT συσκευές που στέλνουν την πληροφορία
- Τον **message broker** που είναι το λογισμικό που διαχειρίζεται τις συνδέσεις μεταξύ των συσκευών και την δρομολόγηση των μηνυμάτων
- Και τους **subscribers** που είναι οι ομάδες ληπτών μηνυμάτων

Από πλευράς ασφάλειας το πρωτόκολλο MQTT υποστηρίζει TLS κρυπτογράφηση, αυθεντικοποίηση με όνομα χρήστη και κωδικό πρόσβασης αλλά και πιστοποιητικά για την κάθε συσκευή αν αυτό απαιτηθεί [14].

2.4.2. Message Broker

Ο message broker χρησιμοποιεί το πρωτόκολλο MQTT με το μοντέλο publish/subscribe υπερτερώντας έναντι άλλων πρωτοκόλλων όπως CoAP, XMPP, AQMP, DDS διότι μπορεί εύκολα να υποστηρίξει χιλιάδες συσκευές που δεν χρειάζεται να είναι άμεσα συνδεδεμένες μεταξύ τους. Αποτελεί τον ενδιάμεσο κρίκο ανάμεσα σε συσκευές που θέλουν να στείλουν και να λάβουν κάποια πληροφορία. Έχει την αποκλειστική διαχείριση και παρακολούθηση των συνδέσεων με τις συσκευές αυξάνοντας έτσι την ασφάλεια στην επικοινωνία. Η ικανότητά του να διαχειρίζεται μηνύματα μεταξύ πολλών συσκευών στηρίζεται στο γεγονός ότι δεν χρησιμοποιεί την διεύθυνση της κάθε συσκευής για να στείλει τα μηνύματα αλλά ονόματα θεμάτων που ονομάζονται topics.

Όπως φαίνεται και στο Σχήμα 2 η διαδικασία ανταλλαγής μηνυμάτων έχει ως εξής. Ένας publisher αφού συνδεθεί με τον broker στέλνει το μήνυμα επιλέγοντας κάποιο topic (π.χ. hua/temp). Ένας subscriber συνδέεται στον broker και εγγράφεται στο topic που τον ενδιαφέρει (π.χ. hua/hum) λαμβάνοντας έτσι τα μηνύματα που τον αφορούν. Η επικοινωνία δεν περιορίζεται σε έναν προς έναν αλλά μπορεί να είναι τύπου ένας publisher προς πολλούς subscribers αλλά και πολλοί publishers προς έναν subscriber. Επίσης όλες οι συσκευές μπορούν να λειτουργήσουν και σαν publisher και σαν subscriber που σημαίνει ότι μπορούν να στείλουν αλλά και να λάβουν μηνύματα.

Μια ακόμα δυνατότητα ενός message broker είναι ότι μπορεί να διατηρεί τα τελευταία μηνύματα του κάθε topic έτσι ώστε αν συνδεθεί ένας subscriber να λάβει το τελευταίο μήνυμα από το topic που τον ενδιαφέρει. Για την μακροπρόθεσμη αποθήκευση των μηνυμάτων ώστε να υπάρχει δυνατότητα αναδρομής στο παρελθόν απαιτείται χρήση βάσης δεδομένων. Για την συγκεκριμένη εργασία έχει επιλεγεί η InfluxDB που περιγράφεται παρακάτω.

2.5. Containers

Τα containers μοιάζουν με εικονικές μηχανές με βασική διαφορά ότι δεν απαιτούν ξεχωριστό λειτουργικό σύστημα. Αντιθέτως μοιράζονται το λειτουργικό σύστημα της μηχανής που τα φιλοξενεί. Χρησιμοποιούνται για να υποστηρίξουν εφαρμογές αξιοποιώντας στο έπακρον τους διαθέσιμους πόρους ενός συστήματος. Βασίζονται στο Namespacing και τα Control Groups για να προσφέρουν την λειτουργικότητα της απομόνωσης. Το namespacing και τα control groups είναι λειτουργικότητες που υποστηρίζονται από το Linux εδώ και χρόνια. Με το namespacing δίνεται η δυνατότητα απομόνωσης διεργασιών ή γκρουπ διεργασιών μεταξύ τους. Για παράδειγμα σε διαφορετικά containers μπορούν να τρέχουν διεργασίες με ίδιο PID. Τα control groups προσφέρουν όρια στα resources που μπορούν να χρησιμοποιήσουν τα γκρουπ των διεργασιών. Μια ακόμα λειτουργικότητα που έχει υιοθετηθεί από τα containers με αποτέλεσμα να κερδίζουν πολύ όσον αφορά τον χώρο που απαιτούν είναι αυτή της διαστρωμάτωσης. Στη διαστρωμάτωση υπάρχει μια φορά το βασικό στρώμα που μπορεί να είναι ένα λειτουργικό σύστημα, μια εφαρμογή και είναι μόνο για ανάγνωση ενώ κάθε αλλαγή δημιουργεί ένα άλλο στρώμα που προστίθεται στα προηγούμενα. Τα παραπάνω χαρακτηριστικά κάνουν ιδανική την χρήση τους για διαφορετικές εφαρμογές που πρέπει να τρέχουν μαζί σε ένα χαμηλών προδιαγραφών υπολογιστικό σύστημα όπως είναι το Raspberry Pi.

2.5.1. Docker

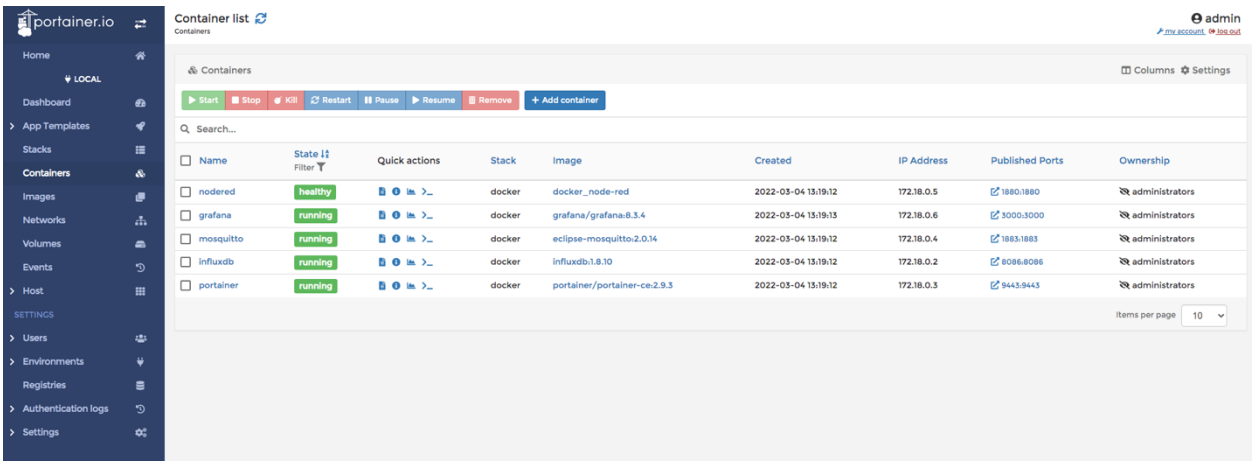
Το Docker είναι ένα λογισμικό ανοιχτού κώδικα που τρέχει σε Linux, Windows και MacOS μηχανές. Χρησιμοποιείται για να δημιουργεί και να διαχειρίζεται containers [15]. Λύνει προβλήματα συμβατότητας που δημιουργούνται κατά την εγκατάσταση εφαρμογών σε διαφορετικού τύπου συστήματα που εξαρτούνται από λογισμικά συγκεκριμένης έκδοσης για να εκτελεστούν και αποτρέπει την δημιουργία προβλημάτων μεταξύ εφαρμογών που δεν είναι συμβατές μεταξύ τους. Παρέχει την λειτουργικότητα της αφαίρεσης αφού κάθε εφαρμογή τρέχει στο δικό της περιβάλλον και δεν επηρεάζεται από την άλλη. Στην συγκεκριμένη μεταπτυχιακή εργασία το docker προσφέρει εύκολη και απροβλημάτιστη εγκατάσταση των διαφορετικών εφαρμογών που απαιτούνται για την υλοποίηση καθώς και διευκόλυνση για οποιαδήποτε μελλοντική επανεγκατάσταση της συγκεκριμένης υλοποίησης.

2.5.2. Docker Compose

Οι περισσότερες σύγχρονες εφαρμογές αποτελούνται από πολλές μικρότερες υπηρεσίες που αλληλοεπιδρούν μεταξύ τους για να σχηματίσουν μια χρήσιμη εφαρμογή. Οι υπηρεσίες αυτές ονομάζονται microservices [15]. Το Docker Compose έρχεται για να λύσει το πρόβλημα της εγκατάστασης και διαχείρισης όλων των microservices που σχηματίζουν μια εφαρμογή. Οι υπηρεσίες και ο τρόπος διασύνδεσης μεταξύ τους ορίζεται μια φορά σε αρχείο YAML το οποίο χρησιμοποιείται στην συνέχεια για την συνολική ανάπτυξη της εφαρμογής με τα απαιτούμενα microservices. Στην παρούσα διπλωματική εργασία αν θεωρηθεί ότι σκοπός της εφαρμογής είναι η αποθήκευση και οπτικοποίηση των ενδείξεων από τους αισθητήρες, τότε τα microservices είναι η InfluxDB, το Grafana και το Telegraf που περιγράφονται παρακάτω.

2.5.3. Portainer

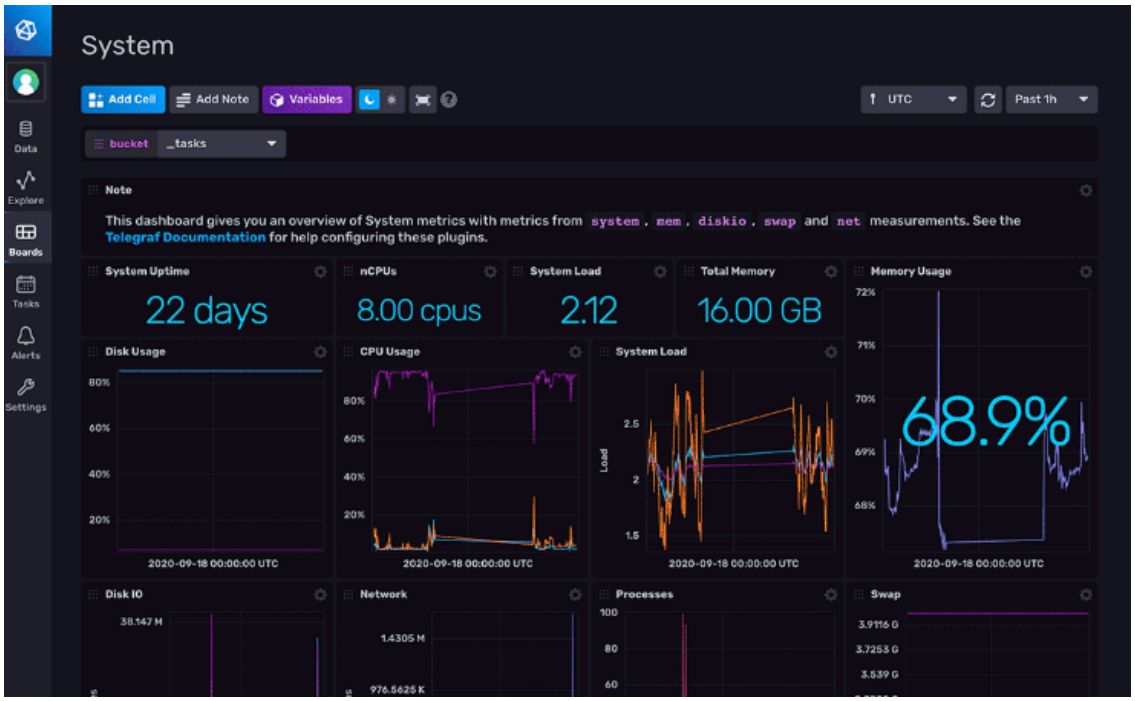
Το Portainer είναι ένα εργαλείο ανοιχτού κώδικα που βοηθάει στην διαχείριση των containers. Είναι ιδανικό για χρήστες που δεν είναι εξοικειωμένοι με την γραμμή εντολών καθώς προσφέρει γραφικό περιβάλλον με δυνατότητες διαχείρισης, ελέγχου και παραμετροποίησης των containers και λοιπών στοιχείων (images, networks, volumes). Το ίδιο το εργαλείο εγκαθίσταται σαν container επιβαρύνοντας ελάχιστα συνολικά το σύστημα. Στην παρακάτω εικόνα φαίνεται το περιβάλλον εργασίας του Portainer με τα containers που εκτελούνται στο raspberry. Μερικές από τις διαθέσιμες ενέργειες που μπορούν να γίνουν είναι να σταματήσει κάποιο container, να δώσει εντολή για restart, να εμφανίσει τα logs ή να αλληλοεπιδράσει με κάποιο container ανοίγοντας γραμμή εντολών.



Εικ. 12. Portainer Overview

2.6. InfluxDB

Η InfluxDB είναι μια βάση δεδομένων χρονοσειρών (time series) ανοιχτού κώδικα χωρίς σχήμα. Είναι γραμμένη στην γλώσσα προγραμματισμού GO και χρησιμοποιείται για την αποθήκευση και ανάκτηση δεδομένων που είναι ευαίσθητα στον χρόνο. Παραδείγματα τέτοιων δεδομένων είναι δεδομένα από αισθητήρες IoT, μετρήσεις για απόδοση εφαρμογών, παρακολούθησης συστημάτων, κ.ά. Προσφέρει δωρεάν αλλά και enterprise έκδοση με δυνατότητες clustering και high availability.



Εικ. 13. Chronograf Dashboard

Συνεργάζεται απόλυτα με το Chronograf που είναι προϊόν της ίδιας εταιρείας και αναλαμβάνει την οπτικοποίηση των δεδομένων αλλά στην εργασία επιλέχθηκε να χρησιμοποιηθεί το Grafana που υποστηρίζει περισσότερες βάσεις δεδομένων και είναι πιο ευέλικτο. Η InfluxDB εκτός από την database engine που ουσιαστικά τρέχει την βάση, παρέχει τον CLI client για την διαχείριση της βάσης και την γλώσσα InfluxQL που μοιάζει με την SQL και δίνει την δυνατότητα εκτέλεσης ερωτημάτων. Σημαντικό επίσης είναι ότι υποστηρίζει πολιτικές διαχείρισης και αυτόματης απόρριψης των παλαιότερων δεδομένων που έχει αποθηκευμένα.

Οι μετρήσεις που αποθηκεύονται στην InfluxDB αποτελούνται από στήλες που ονομάζονται tag keys και field keys και γεμίζουν με τιμές που ονομάζονται tag values και field values. Τα tag keys

περιέχουν κάποιο γνώρισμα της μέτρησης, για παράδειγμα (host, name, machine, color) ενώ τα tag values αλφαριθμητικές τιμές όπως (node1, app2, red, κτλ.). Τα field keys περιέχουν το όνομα της μέτρησης όπως (usage_user, uptime, free) ενώ τα field values την τιμή της μέτρησης για παράδειγμα (0.7, 2, 3.4, κτλ.)

Είναι μια βάση που καλύπτει πλήρως τις ανάγκες της εργασίας διότι υπάρχουν δεδομένα και μετρήσεις από αισθητήρες που είναι ευαίσθητα όσον αφορά τον χρόνο και την ταχύτητα ανάγνωσης και εγγραφής τους.

2.7. Telegraf

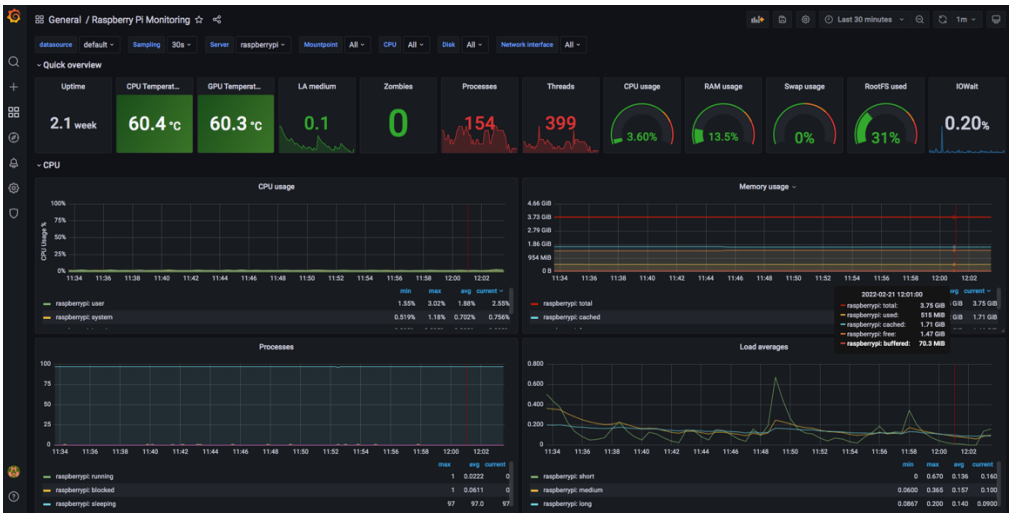
Πρόκειται για έναν ανοιχτού κώδικα agent που είναι και αυτός γραμμένος σε γλώσσα προγραμματισμού GO και χρησιμοποιείται για συλλογή και αναφορά μετρήσεων. Απαιτεί ελάχιστους πόρους για να τρέξει και αυτό το κάνει ιδανικό για παρακολούθηση μικρών IoT συσκευών μέχρι μεγάλων εμπορικών συστημάτων. Το telegraf έχει τη δυνατότητα να διαβάζει δεδομένα από διαφορετικού τύπου πηγές όπως (MQTT, SNMP, HTTP Listener, κ.ά.) και να τα γράφει σε βάσεις και υπηρεσίες όπως (InfluxDB, Graphite, Kafka, MQTT, κ.ά.).

Είναι ένα σύστημα βασισμένο σε τέσσερα διαφορετικού τύπου πρόσθετα (plugins) που αποτελούνται από input, processor, aggregator και output plugins. Τα δεκάδες έτοιμα input plugins που έχει προεγκατεστημένα χρησιμοποιούνται για να διαβάζει διαφορετικού τύπου δεδομένα από διάφορες συσκευές ενώ τα output plugins για να γράφει τα δεδομένα αυτά σε διάφορους προορισμούς. Επίσης υπάρχει η δυνατότητα για έναν χρήστη αν δεν είναι ικανοποιημένος με τα υπάρχοντα input και output plugins να δημιουργήσει τα δικά του. Τα processor και aggregator plugins χρησιμοποιούνται για να διαμορφώσουν και να φιλτράρουν τα δεδομένα πριν τελικά σταλούν στο output plugin.

Στην παρούσα εργασία ένας telegraf agent έχει εγκατασταθεί στο raspberry με σκοπό την παρακολούθηση και αποστολή μετρήσεων στην InfluxDB. Το ποσοστό χρήσης του επεξεργαστή, η κατάσταση της μνήμης, ο υπολειπόμενος ελεύθερος αποθηκευτικός χώρος στην κάρτα SD του raspberry αλλά και οι θερμοκρασίες από τους ενσωματωμένους αισθητήρες είναι μερικές από τις μετρήσεις που θα αποστέλλονται στην InfluxDB. Επιπλέον ο agent θα δημιουργεί την βάση στην οποία και θα εγγράφονται τα δεδομένα.

2.8. Grafana

Το Grafana είναι ένα λογισμικό ανοιχτού κώδικα με βασική του λειτουργία την οπτικοποίηση δεδομένων. Έχει την δυνατότητα να δημιουργεί γραφήματα και dashboards από δεδομένα που χρήζουν παρακολούθησης. Τα δεδομένα αυτά μπορεί να είναι μετρήσεις (metrics) ή και αρχεία καταγραφής (logs) από κάποιον server, εικονική μηχανή ή IoT συσκευή με αισθητήρες. Εκτός από την δωρεάν έκδοση για τοπική εγκατάσταση, προσφέρεται και σαν υπηρεσία στο cloud με κάποιους περιορισμούς αλλά και enterprise έκδοση με περισσότερα πρόσθετα.



Εικ. 14. Grafana Dashboard

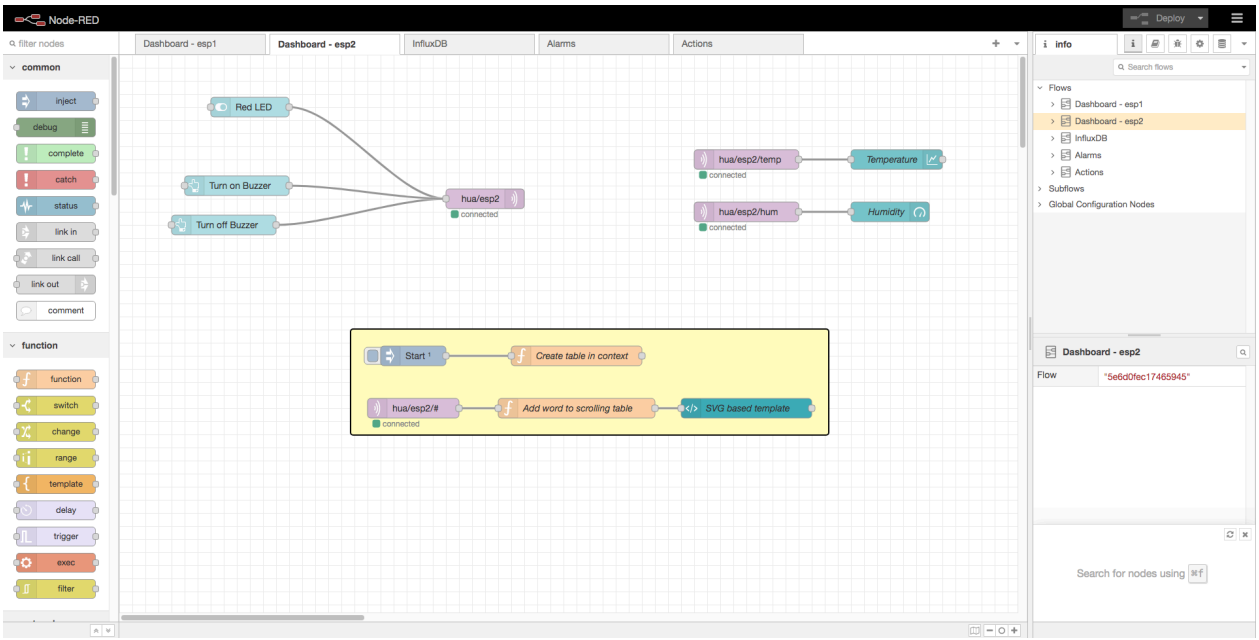
Για την υποδομή του Grafana χρειάζεται τουλάχιστον μια πηγή που θα παράγει metrics και logs τα οποία στη συνέχεια θα αποθηκεύονται σε μια βάση δεδομένων. Το Grafana υποστηρίζει ένα πλήθος από βάσεις όπως (Prometheus, InfluxDB, MySQL, κτλ.) Για την οπτικοποίηση των δεδομένων το Grafana ανακτά τα δεδομένα με μορφή ερωτημάτων από την βάση και δημιουργεί γραφήματα τα οποία και τοποθετούνται σε προκαθορισμένες θέσεις πάνω στο dashboard όπως φαίνεται και στην εικόνα 10.

Στην παρούσα μεταπτυχιακή εργασία το Grafana θα χρησιμοποιηθεί για την απεικόνιση των δεδομένων από τους αισθητήρες σε ένα dashboard. Εκτός από τους αισθητήρες θα φαίνεται και η κατάσταση των βασικών λειτουργιών του raspberry, όπως είναι η χρήση του επεξεργαστή και της μνήμης, η θερμοκρασία λειτουργίας του κ.ά.

2.9. Node-RED

Το Node-RED είναι ένα γραφικό εργαλείο προγραμματισμού ανοιχτού κώδικα που χρησιμοποιείται για την διασύνδεση εφαρμογών, βάσεων δεδομένων, cloud υπηρεσιών, συσκευών IoT κ.ά. Παρέχει με γραφικό τρόπο έτοιμα κομμάτια κώδικα που ονομάζονται nodes και προσφέρουν διάφορες λειτουργικότητες, ενώ τα nodes διασυνδέονται μεταξύ τους προκειμένου να εκτελέσουν μια εργασία. Τα διασυνδεδεμένα αυτά nodes ονομάζονται flows. Ακολουθεί το μοντέλο προγραμματισμού flow-based που σημαίνει ότι υπάρχει ροή δεδομένων μεταξύ των nodes.

Μερικά από τα πλεονεκτήματα που έχει είναι ότι είναι εύκολο στη χρήση ενώ μέσω των nodes που προσφέρει αποκρύπτει την πολυπλοκότητα του προγραμματισμού με αποτέλεσμα ένας χρήστης να ασχολείται περισσότερο με την δημιουργία της λειτουργικότητας που θέλει παρά με την ανάπτυξη του κώδικα.



Εικ. 15. Node-RED Environment

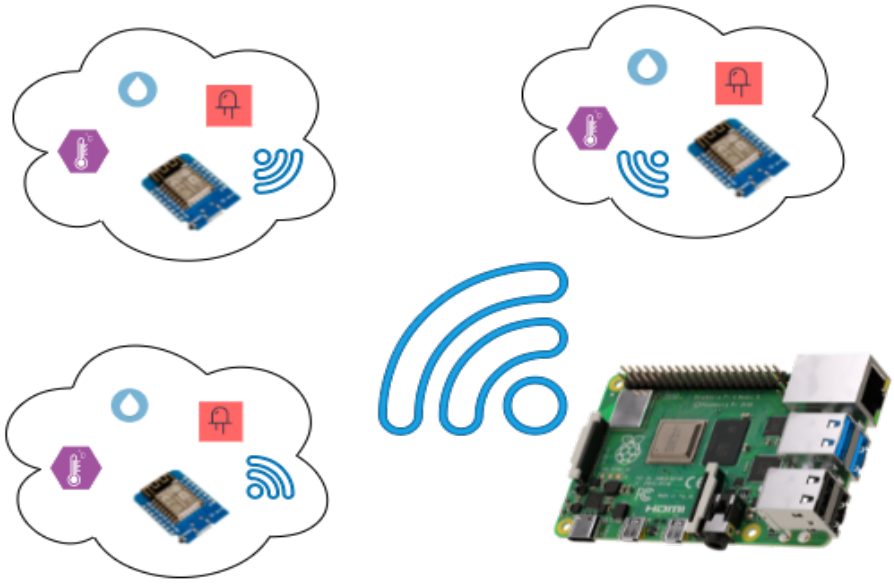
Ο λόγος που χρησιμοποιήθηκε στην εργασία είναι ότι προσφέρει ένα πολύ ελαφρύ περιβάλλον ανάπτυξης και εκτέλεσης, που το καθιστά εξαιρετικό για τη δημιουργία εφαρμογών που πρέπει να είναι γρήγορες και εύκολα παραμετροποιήσιμες, ώστε να μπορούν να εκτελούνται σε υλικό χαμηλού κόστους όπως είναι το Raspberry Pi.

ΚΕΦ.3: ΠΑΡΑΜΕΤΡΟΠΟΙΗΣΗ ΣΥΣΤΗΜΑΤΟΣ

Σε αυτό το κεφάλαιο θα γίνει η παραμετροποίηση του συστήματος κάνοντας χρήση της ansible και ο προγραμματισμός του ESP8266 με sketch αναφοράς περιγράφοντας παράλληλα τον κώδικα που χρησιμοποιήθηκε. Ο κώδικας αποτελείται από C++ για των προγραμματισμό των πλακετών, yaml αρχεία για την ansible και JavaScript για το Node-RED. Η παραμετροποίηση ξεκινάει με την εγκαθίδρυση ασύρματου δικτύου αισθητήρων.

3.1. Wireless Sensor Network

Για λόγους ασφαλείας και για να αποφευχθούν τυχόν προβλήματα απόκρισης και αξιοπιστίας του δικτύου θα δημιουργηθεί ένα ασύρματο δίκτυο αισθητήρων. Οι διασυνδεδεμένες συσκευές που θα ανταλλάσσουν πληροφορίες σε αυτό το δίκτυο είναι το raspberry και οι πλακέτες ESP8266. Σκοπός είναι το δίκτυο αυτό να μην έχει πρόσβαση στο διαδίκτυο ενώ ιδανικά για την διασύνδεση των αισθητήρων θα πρέπει να έχει προηγηθεί η δήλωση των MAC διευθύνσεων των πλακετών ESP8266. Η ασύρματη κάρτα δικτύου του raspberry με το πρότυπο 802.11g θα χρησιμοποιηθεί σαν access point ώστε να συνδεθούν σε αυτήν οι πλακέτες ESP8266. Για την επίτευξη αυτού του στόχου θα πρέπει να γίνει εγκατάσταση των πακέτων hostapd και dnsmasq. Η διαδικασία της δημιουργίας του ασύρματου δικτύου αισθητήρων έχει αυτοματοποιηθεί σε playbook που περιγράφεται παρακάτω.



Σχ. 3. Wireless Sensor Network

3.1.1. hostapd

Πρόκειται για ένα λογισμικό που επιτρέπει σε μια κάρτα δικτύου αν το υποστηρίζει να λειτουργεί σαν access point αλλά και σαν διακομιστής αυθεντικοποίησης. Το αρχείο με τις ρυθμίσεις του λογισμικού βρίσκεται στη διαδρομή /etc/hostapd/hostapd.conf. Εκεί μεταξύ άλλων ορίζεται η κάρτα δικτύου που θα χρησιμοποιηθεί σαν access point, το πρότυπο διασύνδεσης, ο τύπος κρυπτογράφησης καθώς και το όνομα και ο κωδικός πρόσβασης του δικτύου. Στη περίπτωση που απαιτείται η δήλωση των MAC διευθύνσεων των συσκευών προτού τους επιτραπεί η σύνδεση τότε πρέπει να προστεθούν στο αρχείο hostapd.conf οι παρακάτω γραμμές:

```
macaddr_acl=1
accept_mac_file=/etc/hostapd/accept
```

Στη συνέχεια θα πρέπει να δημιουργηθεί αρχείο /etc/hostapd/accept με περιεχόμενο τις MAC διευθύνσεις των ESP8266. Τέλος υπάρχει η δυνατότητα και για προσθήκη αρχείου με μη επιτρεπόμενες MAC διευθύνσεις [16] [17].

3.1.2. dnsmasq

Είναι ένα λογισμικό χαμηλών απαιτήσεων που μπορεί να προσφέρει υπηρεσίες DNS και DHCP σε ένα μικρό δίκτυο συσκευών. Σαν διακομιστής DHCP θα ρυθμιστεί ώστε να εκχωρεί IP διευθύνσεις στις πλακέτες ESP8266 που θα συνδέονται στο ασύρματο δίκτυο αισθητήρων. Σαν διακομιστής DNS έχει την δυνατότητα να απαντάει σε ερωτήματα DNS από την τοπική μνήμη του ή να τα προωθεί σε προ ρυθμισμένους διακομιστές DNS. Το αρχείο με τις ρυθμίσεις του λογισμικού βρίσκεται στη διαδρομή /etc/dnsmasq.conf. Μερικές από τις παραμετροποιήσεις που δύναται να οριστούν σε αυτό το αρχείο είναι το όνομα της κάρτας δικτύου που θα εφαρμόζει τις υπηρεσίες του, το εύρος των διευθύνσεων και ο χρόνος μίσθωσης των IP διευθύνσεων που θα εκχωρεί στις πλακέτες καθώς και ο τρόπος που θα γίνεται η αναζήτηση μιας IP διεύθυνσης (domain-needed) [16] [17].

3.2. Sketches

Στις επόμενες παραγράφους θα περιγραφεί αναλυτικά η δομή των Sketches που υλοποιήθηκαν.

3.2.1. Load Libraries

Όλα τα sketches ξεκινάνε φορτώνοντας τις απαιτούμενες βιβλιοθήκες για το εκάστοτε project. Η βιβλιοθήκη ESP8266WiFi χρησιμοποιείται για την διασύνδεση του ESP8266 σε ένα WIFI δίκτυο. Η βιβλιοθήκη PubSubClient για την υποστήριξη και ανταλλαγή MQTT μηνυμάτων, ενώ η βιβλιοθήκη DHT για την υποστήριξη των DHT αισθητήρων θερμοκρασίας υγρασίας [18] [19].

```
// Load Libraries
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <DHT.h>
```

3.2.2. Defining Variables

Στη συνέχεια δηλώνονται οι μεταβλητές που απαιτούνται. Δηλώνεται ο τύπος του αισθητήρα θερμοκρασίας υγρασίας που επιλέχθηκε καθώς και ο αριθμός του GPIO pin που είναι συνδεδεμένος. Οι λαμπτήρες led και οι υπόλοιποι αισθητήρες δηλώνονται ομοίως.

```
// GPIO PINS
#define DHTTYPE DHT11
#define DHT_SENSOR D5
#define LED_RED D3
#define LED_GREEN D4
#define LED_ORANGE D8
#define BUZZER D6
#define RAIN_SENSOR D7
#define MOTION_SENSOR D1
#define DOOR D2
```

Σε αυτό το σημείο δηλώνεται το WIFI δίκτυο που πρέπει να συνδεθεί η πλακέτα ESP8266. Αν είναι το ασύρματο δίκτυο αισθητήρων που έχει δημιουργήσει το ίδιο το raspberry τότε το ssid και password πρέπει να συμπίπτει με αυτό που θα δηλωθεί στο αρχείο ρυθμίσεων hostapd.conf της ομώνυμης υπηρεσίας.

```
// WIFI Credentials
const char* ssid = "pinet";
const char* password = "12345678";
```

Εδώ δηλώνεται η IP διεύθυνση του MQTT broker. Αφού ο broker τρέχει στο raspberry η IP διεύθυνση θα είναι του ίδιου του raspberry.

```
// MQTT Broker IP address
const char* mqtt_server = "192.168.7.94";
```

Στη συνέχεια δηλώνεται το topic που θα κάνει subscribe η πλακέτα ESP8266 όπως επίσης και τα topics που θα κάνει publish τις μετρήσεις από τους αισθητήρες και την κατάστασή τους.

```
// Topic for ESP to Subscribe
const char* ESP_TOPIC = "hua/esp1";

// Create topic names to Publish
const char* Temperature_Topic = "hua/esp1/temp";
const char* Humidity_Topic = "hua/esp1/hum";
const char* Water_Topic = "hua/esp1/water";
const char* Motion_Topic = "hua/esp1/motion";
const char* Door_Topic = "hua/esp1/door";
```

Εδώ γίνεται η αρχικοποίηση του ESP8266 σαν πελάτη WIFI αλλά και σαν πελάτη MQTT μηνυμάτων.

```
// Initialize espClient and create three variables
WiFiClient espClient;
PubSubClient client(espClient);
```

Αρχικοποιείται ο DHT αισθητήρας θερμοκρασίας υγρασίας.

```
// Initialize DHT Sensor
DHT dht(DHT_SENSOR, DHTTYPE);
```

Δημιουργείται η μεταβλητή now που κρατάει τον χρόνο ενώ αρχικοποιούνται οι υπόλοιπες μεταβλητές που θα χρησιμοποιηθούν στην συνέχεια για να ορίσουν τον ρυθμό επανάληψης των μετρήσεων.

```
// Timers auxiliar variables
long now = millis();
long Last_DHT_Measure = 0;
long Last_Water_Measure = 0;
long Last_Motion_Measure = 0;
long Last_Door_Measure = 0;
```

Σε αυτό το σημείο ολοκληρώνεται η δήλωση και αρχικοποίηση των μεταβλητών και το Sketch συνεχίζει με τον ορισμό των συναρτήσεων ξεκινώντας από την `setup_wifi()` [18] [19].

3.2.3. WiFi() function

Η συνάρτηση αυτή εγκαθιδρύει την ασύρματη σύνδεση του ESP8266 με το raspberry. Παράλληλα εμφανίζει στην οθόνη πληροφορίες όπως το όνομα του δικτύου που προσπαθεί να συνδεθεί και την IP διεύθυνση που εκχωρήθηκε στην πλακέτα αφού γίνει η σύνδεση [18] [19].

```
// Connect to WiFi network
void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("WiFi connected - ESP IP address: ");
  Serial.println(WiFi.localIP());
}
```

3.2.4. Callback() function

Η callback() χρησιμοποιείται για την μετατροπή των MQTT μηνυμάτων σε κατάλληλη μορφή ώστε σύμφωνα με αυτά να εκτελεστεί η αλλαγή στην κατάσταση των αντίστοιχων GPIO pins. Σε αυτή την συνάρτηση μπορεί να γίνει παραμετροποίηση των ενεργειών που εκτελούνται σε σχέση με το μήνυμα που λαμβάνεται. Για παράδειγμα αν έρθει μήνυμα «alert on» να δίνεται εντολή για να ανάψει το κόκκινο led και να χτυπήσει το buzzer. [18] [19]

```
// Receives MQTT messages to a topic that ESP is subscribed
// Prints in serial monitor the message and switches GPIO states if message received
void callback(String topic, byte *payload, unsigned int length) {
    Serial.print("Message arrived in topic: ");
    Serial.println(topic);
    Serial.print("Message:");
    String message;
    for (unsigned int i = 0; i < length; i++) {
        message = message + (char) payload[i];  // convert *byte to string
    }
    Serial.print(message);
    if (message == "red on") { digitalWrite(LED_RED, HIGH); }
    if (message == "red off") { digitalWrite(LED_RED, LOW); }
    if (message == "green on") { digitalWrite(LED_GREEN, HIGH); }
    if (message == "green off") { digitalWrite(LED_GREEN, LOW); }
    if (message == "orange on") { digitalWrite(LED_ORANGE, HIGH); }
    if (message == "orange off") { digitalWrite(LED_ORANGE, LOW); }
    if (message == "buzzer on") { digitalWrite(BUZZER, HIGH); }
    if (message == "buzzer off") { digitalWrite(BUZZER, LOW); }
    Serial.println();
    Serial.println("-----------------------");
}
```

3.2.5. Reconnect() function

Η reconnect() προσφέρει δύο λειτουργικότητες. Η πρώτη είναι η διασύνδεση της πλακέτας με τον broker και η δεύτερη είναι για να κάνει subscribe το ESP8266 με το topic που έχει επιλεγεί ώστε να λαμβάνει τα MQTT μηνύματα που το αφορούν [18] [19].

```
// If ESP can't establish an MQTT communication with broker, keeps trying to reconnect
every 5 seconds
void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Create a random client ID
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);
    // Attempt to connect
    if (client.connect(clientId.c_str())) {
      Serial.println("connected");
     //once connected to MQTT broker, subscribe
      client.subscribe(ESP_TOPIC);
      //client.subscribe("topic2"); I can subscribe to more than one topics
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000); }
    }
}
```

3.2.6. Setup() function

Η βασική συνάρτηση που τρέχει το ESP8266 κατά την εκκίνηση ή επανεκκίνηση του είναι η setup(). Η βασική λειτουργικότητα που προσφέρει είναι να ορίσει τα GPIO pins αν θα χρησιμοποιηθούν σαν είσοδο ή έξοδο καθώς και να καλέσει άλλες συναρτήσεις που θα συνδέσουν την πλακέτα στο δίκτυο και στον MQTT broker [18] [19].

```
void setup() {

  dht.begin();

  Serial.begin(115200); // Start serial communication
  setup_wifi();  // Calls setup_wifi() function to connect to wifi
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
  pinMode(LED_RED, OUTPUT);
  pinMode(LED_GREEN, OUTPUT);
  pinMode(LED_ORANGE, OUTPUT);
  pinMode(BUZZER,OUTPUT);
  pinMode(RAIN_SENSOR,INPUT);
  pinMode(MOTION_SENSOR,INPUT);
  pinMode(DOOR,INPUT);
  digitalWrite(LED_GREEN, HIGH);
}
```

3.2.7. Loop() function

Η τελευταία και πιο σημαντική συνάρτηση είναι η loop(). Είναι η συνάρτηση που εκτελείται και επαναλαμβάνεται καθ' όλη την διάρκεια λειτουργίας του ESP8266. Ξεκινάει ελέγχοντας αν υπάρχει σύνδεση με τον broker ενώ στη συνέχεια κάνει publish τις μετρήσεις και ενδείξεις που λαμβάνει από τους αισθητήρες στα αντίστοιχα topics. Για τον κάθε αισθητήρα υπάρχει ξεχωριστό κομμάτι κώδικα που φαίνεται σχολιασμένο στον κώδικα. Παραμετροποιήσεις που μπορούν να γίνουν εδώ αφορούν την συχνότητα των μετρήσεων αλλάζοντας τα δευτερόλεπτα στην δομή επανάληψης του εκάστοτε αισθητήρα. Επίσης μπορεί να γίνει αλλαγή του μηνύματος που θα γίνεται publish στο εκάστοτε topic αφού αναγνωσθεί η κατάσταση του αισθητήρα. Για παράδειγμα αντί να στέλνει «door off» να στέλνει «door closed» [18] [19].

```
void loop() {

  if (!client.connected()) { // Check if ESP is connected to the MQTT broker. If not
reconnect.
    reconnect();
  }
  client.loop();

  if(!client.loop())
    client.connect("ESP8266Client");

  now = millis();
  // Publishes new temperature and humidity every 30 seconds
  if (now - Last_DHT_Measure > 30000) {
    Last_DHT_Measure = now;
    float h = dht.readHumidity();
    float t = dht.readTemperature();

    // Check if any reads failed and exit early (to try again).
    if (isnan(h) || isnan(t)) {
      Serial.println("Failed to read from DHT sensor!");
      return;
    }

    // Convert float to string
    static char temperatureTemp[7];
    dtostrf(t, 6, 2, temperatureTemp);


    static char humidityTemp[7];
    dtostrf(h, 6, 2, humidityTemp);

    // Publishes Temperature and Humidity values
    client.publish(Temperature_Topic, temperatureTemp);
    client.publish(Humidity_Topic, humidityTemp);

    Serial.print("Humidity: ");
    Serial.print(h);
    Serial.print(" %\t Temperature: ");
    Serial.print(t);
    Serial.print(" *C ");
```

```
    Serial.println();
  }

  // Detects for water every 3 seconds
  if (now - Last_Water_Measure > 3000) {
    Last_Water_Measure = now;

    if (digitalRead(RAIN_SENSOR)==LOW)
    {
      Serial.println("Water Detected!!!");
      client.publish(Water_Topic, "water");
      digitalWrite(BUZZER,HIGH);
    }
    else
    {
      // Serial.println("No Water");
      client.publish(Water_Topic, "no water");
      digitalWrite(BUZZER,LOW);
    }
  }

  // Detects motion every 2 seconds
  if (now - Last_Motion_Measure > 2000) {
    Last_Motion_Measure = now;

    int val = digitalRead(MOTION_SENSOR);
    if (val == HIGH)
    {
      Serial.println("Motion Detected!!!");
      client.publish(Motion_Topic, "motion");
      digitalWrite(LED_ORANGE, HIGH);
    }
    else
    {
      // Serial.println("No Motion detected!");
      client.publish(Motion_Topic, "no motion");
      digitalWrite(LED_ORANGE, LOW);
    }

  }

  // Detects door status every second
  if (now - Last_Door_Measure > 1000) {
    Last_Door_Measure = now;

    if (digitalRead(DOOR)==HIGH)
    {
      // Serial.println("Door is Closed!");
      client.publish(Door_Topic, "door off");
      digitalWrite(LED_RED, LOW);
    }
    else
    {
      Serial.println("Door is Open!");
      client.publish(Door_Topic, "door on");
      digitalWrite(LED_RED, HIGH);
    }

  }
}
```

3.3. Playbooks

Στις επόμενες παραγράφους θα γίνει αναλυτική περιγραφή της δομής των playbooks που υλοποιήθηκαν. Ένα playbook μεταξύ άλλων αποτελείτε από tasks και handlers. Το κάθε task εκτελεί κάποιες ενέργειες ενώ ο handler τρέχει στο τέλος του playbook αφού τελειώσουν όλα τα tasks και εφόσον υπάρχει αλλαγή στο task που το κάλεσε.

3.3.1. docker-install.yml

Το συγκεκριμένο playbook ξεκινάει απενεργοποιώντας για λόγους ασφαλείας την δυνατότητα απομακρυσμένης σύνδεσης στο raspberry μέσω SSH με κωδικό πρόσβασης. Επίσης καλεί τον handler restart sshd για να εκτελέσει restart στην υπηρεσία του SSH εφόσον πραγματοποιηθεί κάποια αλλαγή στο configuration αρχείο.

```
- name: Disable SSH Password Authentication
  lineinfile:
    dest: /etc/ssh/sshd_config
    regexp: "^(#\s*)?PasswordAuthentication yes"
    line: "PasswordAuthentication no"
    state: present
  notify:
- restart sshd
```

Στη συνέχεια γίνεται αναβάθμιση της λίστας με τα διαθέσιμα πακέτα χρησιμοποιώντας τον διαχειριστή πακέτων apt καθώς και η αναβάθμιση των ήδη εγκατεστημένων πακέτων που υπάρχουν στο raspberry.

```
 - name: Update apt repo and cache
   apt: update_cache=yes force_apt_get=yes cache_valid_time=3600
 - name: Upgrade all packages
   apt: upgrade=dist force_apt_get=yes
```

Στο Linux αν για την εγκατάσταση κάποιου πακέτου απαιτηθεί επανεκκίνηση του συστήματος τότε εγγράφεται ένα αρχείο με όνομα reboot-required στο παρακάτω path. Η ansible ελέγχει την ύπαρξη αυτού του αρχείου και εκτελεί επανεκκίνηση μόνο στην περίπτωση που βρεθεί το συγκεκριμένο αρχείο .

```
 - name: Check if a reboot is needed
   register: reboot_required_file
   stat: path=/var/run/reboot-required
 - name: Reboot if kernel updated
   reboot:
   when: reboot_required_file.stat.exists
```

Σε αυτό το task εγκαθίσταται μια σειρά από απαιτούμενα πακέτα για την λειτουργία του docker και του docker-compose.

```
    - name: Install a list of packages
      apt:
        name: [libffi-dev, libssl-dev, python3, python3-pip, vim]
```

Σε αυτό το κομμάτι του playbook γίνεται download και εκτελείται το script που κάνει την εγκατάσταση του docker αν δεν υπάρχει ήδη. Επίσης δίνονται τα απαραίτητα δικαιώματα στον χρήστη pi ώστε να μπορεί να τρέχει εντολές docker.

```
    - name: Get docker intallation script
      shell: curl -fsSL https://get.docker.com -o get-docker.sh
      args:
        creates: /home/pi/get-docker.sh
    - name: Install docker
      shell: sh /home/pi/get-docker.sh
      args:
        creates: /usr/bin/docker
    - name: Add pi user to docker group
      user: name=pi groups=docker append=yes
```

Τέλος γίνεται η αναβάθμιση του πακέτου pip και η εγκατάσταση του docker-compose αφού γίνει download από το github το εκτελέσιμο αρχείο.

```
    - name: Upgrade pip
      pip:
        name: pip
        executable: pip3
        state: latest

    - name: Install docker-compose
      command:
        cmd: "{{ item }}"
      with_items:
        - curl -L
"https://github.com/docker/compose/releases/download/v2.2.2/docker-compose-linux-
armv7" -o /usr/bin/docker-compose
        - chmod +x /usr/bin/docker-compose
```

Ο handler με όνομα restart sshd όταν καλείται εκτελεί restart στην υπηρεσία του SSH.

```
  handlers:
    - name: restart sshd
      service:
        name: sshd
        state: restarted
```

3.3.2. sensor-network.yml

Με αυτό το playbook θα γίνει η εγκατάσταση και παραμετροποίηση του δικτύου αισθητήρων που θα χρησιμοποιηθεί για την ασύρματη διασύνδεση του raspberry με τις πλακέτες ESP8266. Το playbook ξεκινάει αναβαθμίζοντας την λίστα με τα διαθέσιμα πακέτα ενώ στη συνέχεια εγκαθιστά τις εφαρμογές dnsmasq και hostapd. Τέλος σταματάει τις υπηρεσίες των εν λόγω εφαρμογών ώστε να γίνει στην συνέχεια η παραμετροποίησή τους.

```
    - name: Update apt repo and cache
      apt: update_cache=yes force_apt_get=yes cache_valid_time=3600
    - name: Upgrade all packages
      apt: upgrade=dist force_apt_get=yes

    - name: Install dnsmasq & hostapd
      apt:
        name: [dnsmasq, hostapd]
    - name: Stop services
      service:
        name: "{{ item }}"
        state: stopped
      with_items:
        - 'dnsmasq'
        - 'hostapd'
```

Στη συνέχεια ξεκλειδώνεται η ασύρματη κάρτα δικτύου του raspberry και ορίζεται στατική διεύθυνση σε αυτή.

```
    - name: Unblock Wireless
      command: rfkill unblock wifi
    - name: Configure a static IP for the wlan0 interface
      blockinfile:
        path: /etc/dhcpcd.conf
        block: |
          interface wlan0
          nohook wpa_supplicant
          static ip_address=192.168.50.1/24
          static routers=192.168.50.1
```

Σε αυτό το task γίνεται η παραμετροποίηση του dnsmasq αντιγράφοντας το σχετικό αρχείο dnsmasq.conf.

```
    - name: Configure the DHCP server (dnsmasq)
      copy:
        src: ../files/dnsmasq.conf
        dest: /etc/dnsmasq.conf
        backup: yes
      notify: restart dnsmasq
```

Σε αυτό το κομμάτι του κώδικα γίνεται η παραμετροποίηση του hostapd και δίνεται εντολή για εκκίνηση των δύο υπηρεσιών dnsmasq και hostapd.

```
    - name: Configure the access point (hostapd) 1/2
      copy:
        src: ../files/hostapd.conf
        dest: /etc/hostapd/hostapd.conf
      notify: restart hostapd
    - name: Configure the access point (hostapd) 2/2
      lineinfile:
        path: /etc/default/hostapd
        line: DAEMON_CONF="/etc/hostapd/hostapd.conf"

    - name: Start & Unmask Services
      service:
        enabled: yes
        masked: no
        name: "{{ item }}"
        state: started
      with_items:
        - 'dnsmasq'
        - 'hostapd'
```

Οι παρακάτω handlers επανεκκινούν τις υπηρεσίες hostapd και dnsmasq εφόσον υπάρξουν αλλαγές στο task που τους κάλεσε.

```
  handlers:
    - name: restart hostapd
      service:
        name: hostapd
        state: restarted
    - name: restart dnsmasq
      service:
        name: dnsmasq
        state: restarted
```

3.3.3. telegraf-install.yml

Με αυτό το playbook εγκαθίσταται στο raspberry το telegraf καθώς και το configuration για την βάση που θα αποθηκεύονται τα metrics κ.ά. Σκοπός του telegraf agent είναι η παρακολούθηση της υγείας του raspberry. Το playbook ξεκινάει φορτώνοντας τις μεταβλητές που θα χρησιμοποιηθούν αργότερα στο template που θα γεμίσει το configuration αρχείο.

```
    - name: Load Configuration Variables
      include_vars: ../config.yml
```

Στη συνέχεια προστίθεται το κλειδί και το repository της influxdata, εγκαθίσταται το telegraf και ξεκινάει η υπηρεσία του.

```
    - name: Add apt key
      apt_key:
        url: https://repos.influxdata.com/influxdb.key
        state: present

    - name: Add repository
      apt_repository:
        repo: deb https://repos.influxdata.com/debian bullseye stable
        state: present

    - name: Install Telegraf
      apt:
        name: telegraf
        state: present
        update_cache: true

    - name: Make sure telegraf is running
      systemd:
        state: started
        name: telegraf
```

Στα επόμενα δύο tasks αντιγράφεται το configuration στο raspberry ενώ δίνονται και τα απαραίτητα δικαιώματα για να μπορεί το telegraf να διαβάζει την θερμοκρασία από την GPU του raspberry.

```
    - name: Copy configuration
      template:
        src: ../templates/telegraf.j2
        dest: /etc/telegraf/telegraf.conf
      notify: Restart telegraf

    - name: Add telegraf to video group
      user:
        name: telegraf
        groups: video
        append: yes
      notify: Restart telegraf
```

Τέλος ο handler Restart telegraf εκτελεί επανεκκίνηση της υπηρεσίας του telegraf όποτε και αν αυτό απαιτηθεί.

```
  handlers:
    - name: Restart telegraf
      systemd:
        name: telegraf
        state: restarted
```

3.3.4. deploy-stack.yml

Πρόκειται για το τελικό playbook το οποίο εκκινεί όλα τα containers με τις εφαρμογές που χρειάζονται για το σύστημα. Ξεκινάει φορτώνοντας τις μεταβλητές και αντιγράφοντας την δομή των φακέλων που θα αποτελέσουν τον μόνιμο αποθηκευτικό χώρο των containers.

```
    - name: Load Configuration Variables
      include_vars: ../config.yml

    - name: Copy docker files
      copy:
        src: ../docker
        dest: /home/pi
        owner: pi
        group: pi
```

Με το πρώτο task αντιγράφεται στο raspberry το create-db.iql αρχείο το οποίο κατά την εκκίνηση της influxdb δημιουργεί την βάση που θα αποθηκεύονται οι μετρήσεις ενώ το δεύτερο task παράγει το .env αρχείο με τις μεταβλητές που θα χρησιμοποιηθούν από το docker-compose.

```
    - name: Copy influxdb initial script variables
      template:
        src: ../templates/create-db.iql.j2
        dest: /home/pi/docker/influxdb/init/create-db.iql
        owner: pi
        group: pi

    - name: Copy docker compose environmental variables
      template:
        src: ../templates/docker.env.j2
        dest: /home/pi/docker/.env
        owner: pi
        group: pi
```

Τέλος εκτελείται η εντολή docker-compose up -d που εκκινεί όλα τα containers.

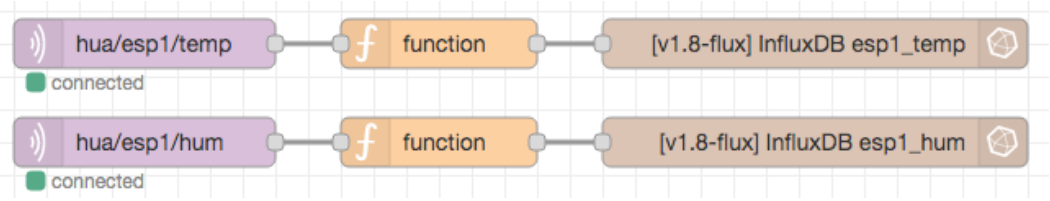
```
    - name: Start containers
      command: docker-compose up -d
      become_user: pi
      args:
        chdir: /home/pi/docker
```

3.4 Node-Red

Στις επόμενες παραγράφους θα γίνει η περιγραφή των παραμετροποιήσεων (flows) που έγιναν στο Node-Red.

3.4.1 Node-Red και InfluxDB

Ο τρόπος που γίνεται η εγγραφή των ενδείξεων από τους αισθητήρες στην βάση φαίνεται στην παρακάτω εικόνα.

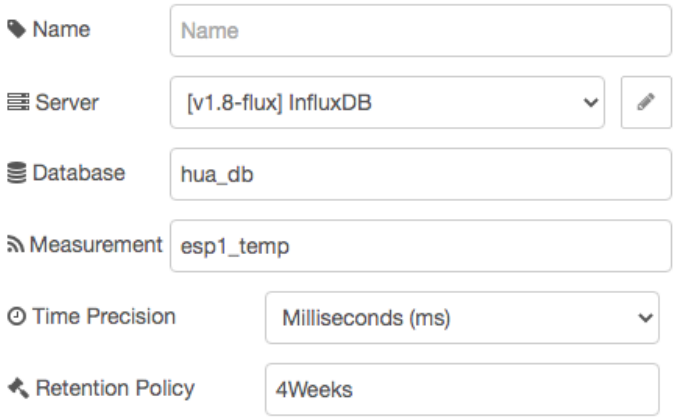


Εικ. 16. Node-Red & InfluxDB

Το flow ξεκινάει με ένα mqtt in node το οποίο έχει κάνει subscribe στο topic hua/esp1/temp. Σε αυτό το topic γίνεται publish η θερμοκρασία που αντιλαμβάνεται ο αισθητήρας DHT11 που είναι συνδεδεμένος με τον μικροελεγκτή esp1. Στη συνέχεια ένα function node μετατρέπει την θερμοκρασία από string σε αριθμό.

```
msg.payload = Number(msg.payload);
return msg;
```

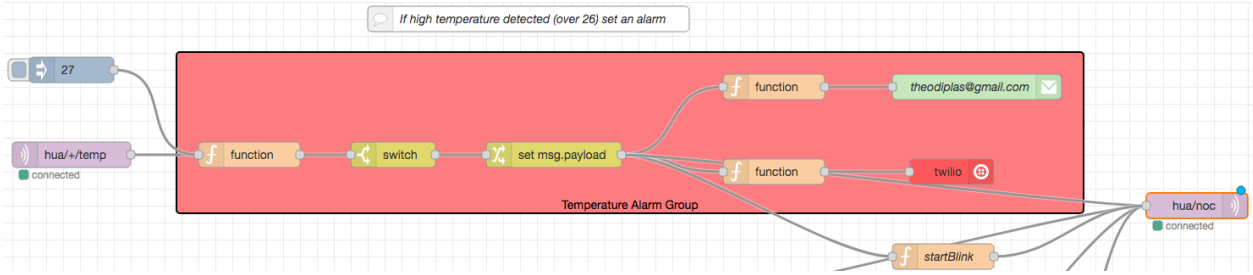
Τέλος ένα influxdb out node γράφει την θερμοκρασία στην βάση hua_db και συγκεκριμένα στο measurement eps1_temp. Το Retention Policy που δηλώνεται εδώ θα πρέπει να συμπίπτει με αυτό της βάσης.



Εικ. 17. InfluxDB out Node

3.4.2 High Temperature Alarm

Το flow που φαίνεται στην παρακάτω εικόνα υλοποιεί την λειτουργία της ειδοποίησης που παρέχει το σύστημα, στην περίπτωση που η θερμοκρασία ξεπεράσει έναν προκαθορισμένο όριο. Η ειδοποίηση γίνεται μέσω email και sms. Το mqtt in node εκμεταλλεύεται τον wildcard χαρακτήρα + ώστε να λάβει ενδείξεις υψηλής θερμοκρασίας από οποιονδήποτε αισθητήρα μέσα στο data center. Στη συνέχεια μέσω μιας function που μετατρέπει την θερμοκρασία από string σε αριθμό γίνεται η σύγκριση με ένα switch node. Αν ξεπεράσει το προκαθορισμένο όριο ενεργοποιείται το επόμενο node που στέλνει ένα απλό μήνυμα **buzzer on** στις functions αλλά και στο mqtt out node.

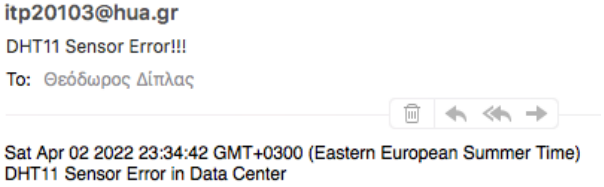


Εικ. 18. Temperature Alarm Flow

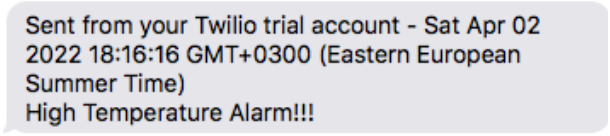
Οι functions συντάσσουν ένα μήνυμα το οποίο αποστέλλεται στα επόμενα δύο nodes.

```
var date =new Date();
var message="";

msg.topic="High Temperature Alarm";
message=" High Temperature Alarm in Data Center ";

msg.payload=date+"\n"+message;
return msg;
```

Το email node στέλνει email σε λογαριασμό που έχει προρυθμιστεί ενώ το twillio out node είναι υπεύθυνο για την αποστολή sms. Το twillio απαιτεί registration στο αντίστοιχο site για να λειτουργήσει παρέχοντας περιορισμένο αριθμό από δωρεάν sms. Παραδείγματα ειδοποιήσεων φαίνονται στις παρακάτω εικόνες.



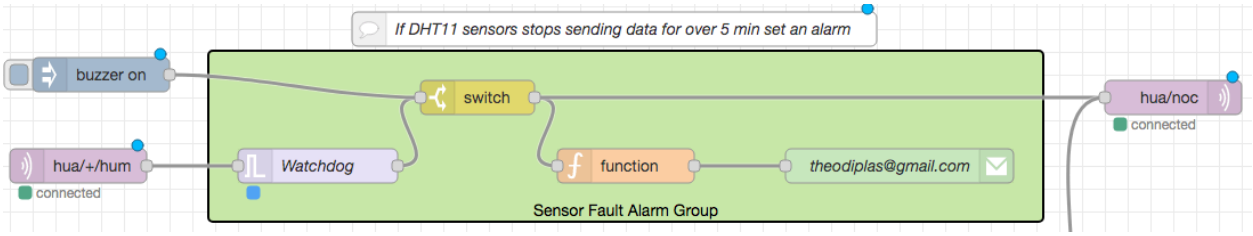
Εικ. 19. Email Alarm Example



Εικ. 20. SMS Alarm Example

3.4.2 Sensor Fault Alarm

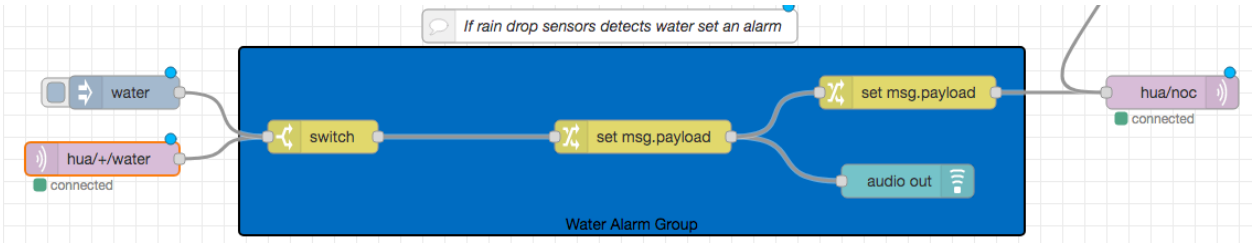
Στόχος αυτού του flow είναι η παρακολούθηση των αισθητήρων θερμοκρασίας και υγρασίας DHT11 και η ενημέρωση με email κάποιου διαχειριστή σε περίπτωση δυσλειτουργίας τους. Η διαδικασία ξεκινάει με ένα mqtt in node το οποίο έχει κάνει subscribe σε όλα τα topics του data center που αφορούν την υγρασία . Στη συνέχεια ένα trigger node ενεργοποιείται αυτόματα αν παρέλθουν 5 λεπτά από την τελευταία ένδειξη υγρασίας που έλαβε. Τέλος όπως και στο προηγούμενο alarm μια function node συντάσσει το μήνυμα το οποίο αποστέλλεται με την συνδρομή του email node.



Εικ. 21. Sensor Fault Alarm Flow

3.4.3 Water Alarm

Το συγκεκριμένο flow παρακολουθεί τον αισθητήρα νερού του συστήματος και στην περίπτωση θετικής ένδειξης ενεργοποιεί το buzzer στέλνοντας αντίστοιχο μήνυμα μέσω ενός mqtt out node. Επίσης έχει προστεθεί ένα audio out node το οποίο εκφωνεί το μήνυμα που θα λάβει αν ο διαχειριστής έχει ανοιχτό το dashboard που περιγράφεται στη συνέχεια.

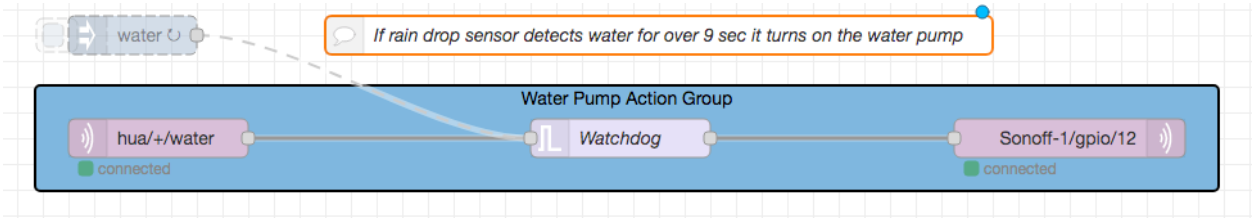


Εικ. 22. Water Alarm Flow

Κλείνοντας με τα alarms γίνεται αντιληπτή η ευελιξία σε παραμετροποιήσεις που προσφέρει το node-red καθώς και η εφαρμογή διαδικασιών χωρίς την ανάπτυξη σημαντικού όγκου κώδικα για τις απαιτήσεις του εκάστοτε σεναρίου χρήσης.

3.4.4 Water Pump Action

Αυτό το flow ενεργοποιεί τον διακόπτη Sonoff σε περίπτωση που κάποιος αισθητήρας νερού αποστείλει συνεχόμενες ενδείξεις για παρουσία νερού. Στην συγκεκριμένη υλοποίηση για χρόνο πάνω από 9 δευτερόλεπτα. Ο διακόπτης ενεργοποιείται αφού λάβει mqtt μήνυμα 1. Η λογική του flow είναι να εκκινήσει μια αντλία νερού αποτρέποντας το ενδεχόμενο της πλημμύρας κάτω από το υπερυψωμένο δάπεδο του data center.

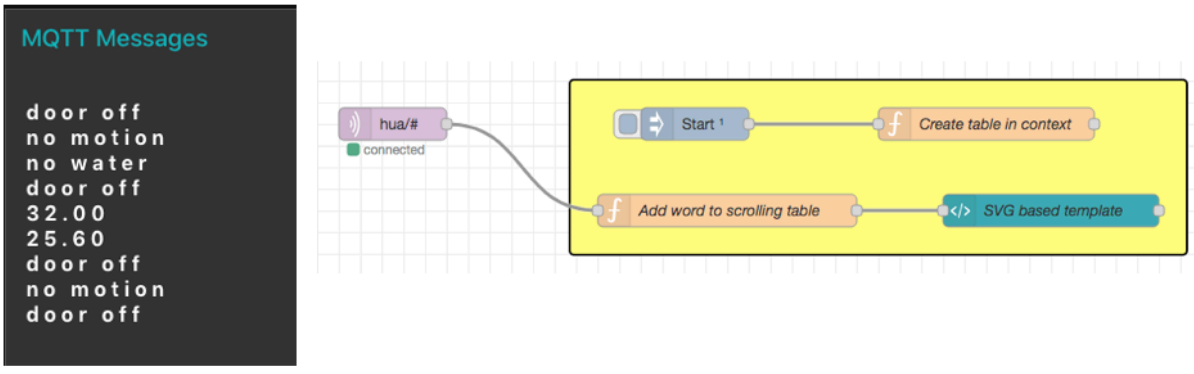


Εικ. 23. Water Pump Action Flow

3.4.5 Dashboard

Το dashboard παραμετροποιήθηκε έτσι ώστε να προσφέρει μια σειρά από λειτουργικότητες στον διαχειριστή όπως έλεγχο των mqtt μηνυμάτων και αντιμετώπιση προβλημάτων στην διασύνδεση των μικροελεγκτών, ενεργοποίηση/απενεργοποίηση της αντλίας νερού, έλεγχος λειτουργικότητας buzzer και leds.

Τα μηνύματα mqtt από το topic που θα επιλεγεί εμφανίζεται στο dashboard με την παρακάτω μορφή ενώ το flow φαίνεται στα δεξιά.



Εικ. 24. MQTT Messages Dashboard & Flow

Ο κώδικας για το scrolling text είναι μια προσφορά του Andrei Ochmat [20].

3.5 Telegraf

Οι παρακάτω παραμετροποιήσεις έγιναν στο **/etc/telegraf/telegraf.conf** configuration αρχείο του telegraf. Metrics που αφορούν τον επεξεργαστή, την μνήμη, το δίκτυο και την θερμοκρασία του raspberry αποστέλλονται μέσω του agent στην influxdb.

```
[[inputs.kernel]]
[[inputs.mem]]
[[inputs.processes]]
[[inputs.swap]]
[[inputs.system]]
[[inputs.net]]
[[inputs.netstat]]

[[inputs.file]]
  files = ["/sys/class/thermal/thermal_zone0/temp"]
  name_override = "cpu_temperature"
  data_format = "value"
  data_type = "integer"

[[inputs.exec]]
  commands = ["/opt/vc/bin/vcgencmd measure_temp"]
  name_override = "gpu_temperature"
  data_format = "grok"
  grok_patterns = ["%{NUMBER:value:float}"]
```

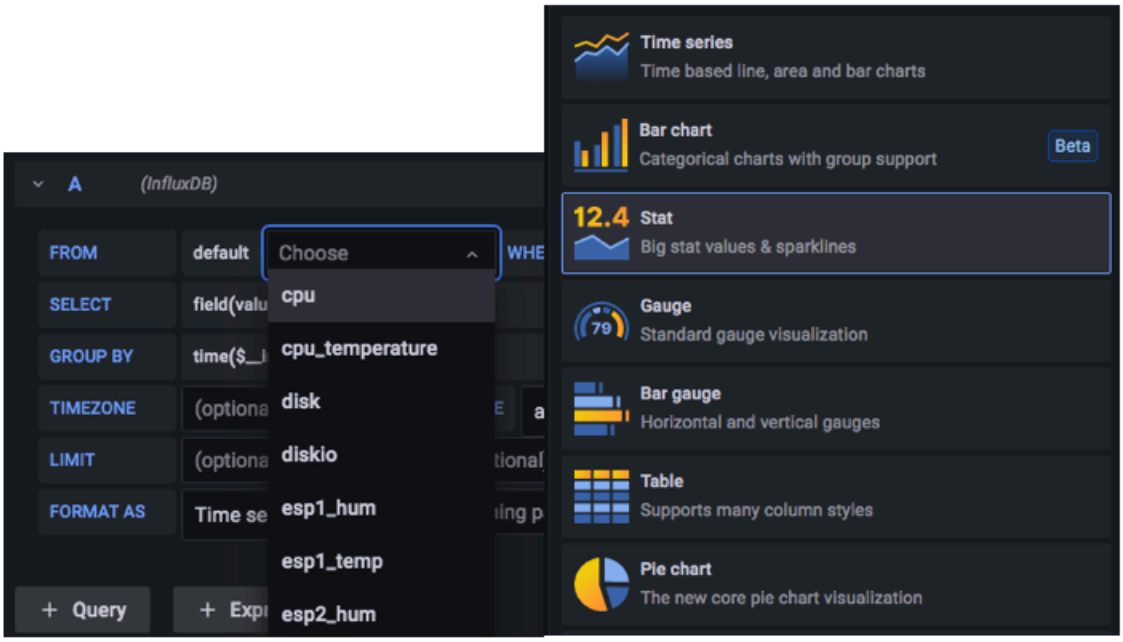
Σε αυτό το σημείο ορίζεται η βάση που θα αποστέλλονται οι ενδείξεις.

```
[[outputs.influxdb]]
 urls = ["http://localhost:8086"]
 username = "hua"
 password = "itp20103"
```

Όλο το configuration του telegraf φορτώνεται αυτόματα στο raspberry μέσω playbook κατά την διάρκεια εγκατάστασης του συστήματος.

3.6 Grafana

Για την οπτικοποίηση των δεδομένων έχει δημιουργηθεί dashboard στο Grafana με όνομα Data Center που υπάρχει στον σύνδεσμο GitHub (https://github.com/theohitman/pms-thesis/tree/main/dashboards). Το Dashboard αποτελείται από panels που είναι πλακίδια με ενδείξεις. Το κάθε panel αποτελείται από ένα query και ένα visualization. Το query ορίζει ποιες ενδείξεις θα εμφανίζει ενώ το visualization ορίζει τον τρόπο που θα εμφανίζονται. Στην παρακάτω εικόνα φαίνεται αριστερά το query ενώ δεξιά ένα μέρος των vizualizations που υποστηρίζει το Grafana.



Εικ. 25. Queries & Visualizations on Grafana

3.7 Docker Compose

Για αβίαστο deployment του software stack του συστήματος έχει δημιουργηθεί ένα αρχείο docker-compose που βρίσκεται στον σύνδεσμο GitHub (https://github.com/theohitman/pms-thesis/tree/main/docker). Βασική παραμετροποίηση που έχει γίνει εδώ είναι το μόνιμο storage των απαραίτητων αρχείων και φακέλων όλων των services. Αυτό επετεύχθη με bind mounts σε μια δομή φακέλων στο raspberry όπως φαίνεται παρακάτω.

```
docker/
├── docker-compose.yaml
├── grafana
│   ├── data
│   └── grafana.ini
├── influxdb
│   ├── data
│   ├── influxdb.conf
│   └── init
├── mosquitto
│   ├── config
│   ├── data
│   └── log
└── nodered
    ├── data
    └── Dockerfile
```

Εικ. 26. Docker Compose Bind Mounts

Η δημιουργία της δομής των φακέλων καθώς και η εκκίνηση του stack γίνεται αυτόματα μέσω playbook κατά την διάρκεια εγκατάστασης του συστήματος.

ΚΕΦ.4: ΣΕΝΑΡΙΟ ΧΡΗΣΗΣ

Στο κεφάλαιο αυτό θα γίνει η εγκατάσταση του συστήματος σε ένα υποθετικό data center. Για λόγους απλότητας το data center θα αποτελείται από ένα και μόνο rack. Το rack είναι μια φυσική δομή που αποτελείται κατά βάση από ένα σιδερένιο πλαίσιο (ικρίωμα) και σκοπός του είναι να φιλοξενεί διακομιστές, δικτυακό εξοπλισμό και καλώδια.

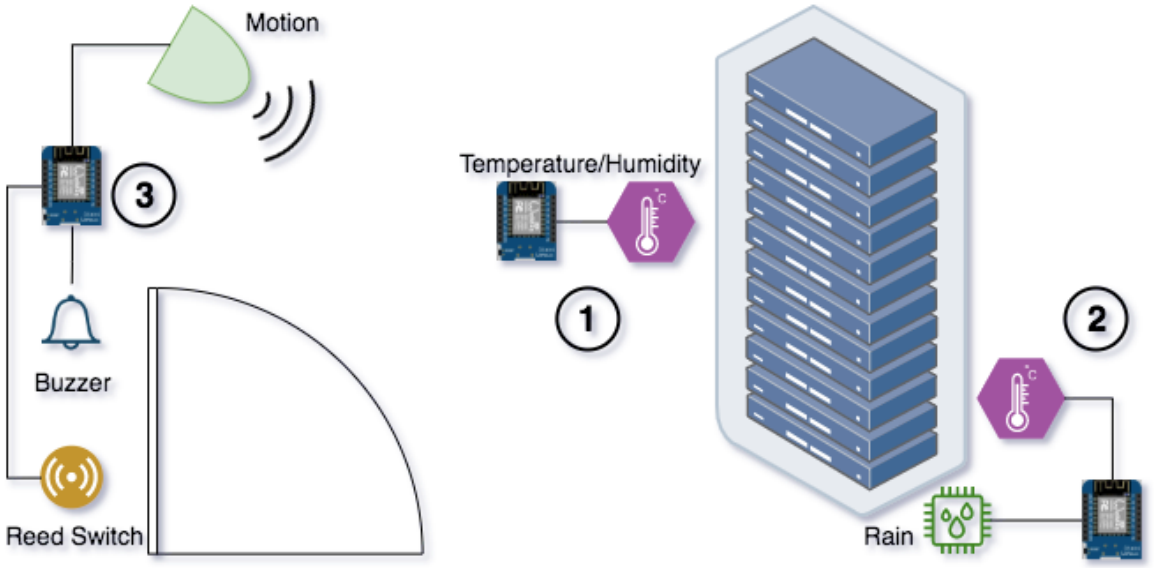
4.1. Περιγραφή

Το σύστημα θα αποτελείται από ένα Raspberry Pi 4 το οποίο θα είναι τοποθετημένο σε οποιοδήποτε σημείο μέσα στο data center ενώ θα πρέπει να έχει πρόσβαση σε παροχή ρεύματος και δικτύου με καλώδιο ethernet. Για την ασύρματη αποστολή μετρήσεων/ενδείξεων θα χρησιμοποιηθούν τέσσερις μικροελεγκτές Wemos D1 mini ESP8266 διασυνδεδεμένοι με αισθητήρες, led και buzzer ως εξής:

1. Ο πρώτος (**ESP1**) θα είναι συνδεδεμένος με έναν αισθητήρα θερμοκρασίας-υγρασίας DHT11 και θα εγκατασταθεί στο μπροστινό μέρος του rack ώστε να υπάρχουν δεδομένα θερμοκρασίας και υγρασίας κατά την είσοδο του αέρα στους διακομιστές. Η τροφοδοσία του μικροελεγκτή θα γίνει από μια ελεύθερη USB θύρα στο μέσον περίπου του rack.
2. Ο δεύτερος (**ESP2**) θα εγκατασταθεί στο οπίσθιο μέρος του rack και προς το πάτωμα, διότι εκτός του αισθητήρα θερμοκρασίας-υγρασίας DHT11, θα έχει συνδεδεμένο και έναν αισθητήρα νερού (Rain Sensor) ώστε να εντοπιστεί τυχόν παρουσία νερού κάτω από το υπερυψωμένο δάπεδο του data center.
3. Ο τρίτος (**ESP3**) μικροελεγκτής θα τοποθετηθεί επάνω από την κεντρική είσοδο του data center. Σκοπός του είναι ελέγχει και να στέλνει ενδείξεις από τον αισθητήρα ανίχνευσης κίνησης (HC-SR501) καθώς και τον μαγνητικό διακόπτη (Reed Switch) που θα είναι άμεσα συνδεδεμένοι. Για την τροφοδοσία του με ρεύμα θα πρέπει να υπάρχει παροχή ρεύματος κοντά στην κεντρική είσοδο.
4. Ο τέταρτος (**NOC**) μικροελεγκτής θα χρησιμοποιηθεί για παρακολούθηση της κατάστασης του συστήματος και του data center. Θα πρέπει να βρίσκεται σε διπλανό δωμάτιο του data center ώστε να υπάρχει ασύρματη σύνδεση με το raspberry ενώ η παροχή τροφοδοσίας του μπορεί να γίνει από πρίζα ρεύματος ή από θύρα USB. Σκοπός

του είναι να παρέχει οπτικές και ακουστικές ενδείξεις στον υπεύθυνο διαχείρισης του data center.

Στο παρακάτω σχήμα παρουσιάζεται η διάταξη των μικροελεγκτών μέσα στο data center. Επιπροσθέτως εμφανίζεται η διασύνδεση του κάθε μικροελεγκτή με τους αισθητήρες που θα ελέγχει.



Σχ. 4. Διάταξη Μικροελεγκτών (ESP1, ESP2, ESP3)

4.2. Υλικά και Κόστος

Τα υλικά και το κόστος αγοράς τους την τρέχουσα χρονική περίοδο, παρουσιάζονται συνοπτικά στον παρακάτω πίνακα. Στο κόστος υλοποίησης του συστήματος δεν υπολογίστηκαν αναλώσιμα και επιμέρους μικρότερα εξαρτήματα (καλώδια, αντιστάσεις, κτλ.).

Υλικό	**Ποσότητα**	**Κόστος (€)**
Raspberry Pi 4 Model B (with microSD, power adapter and case)	1	100
WeMos D1 mini ESP8266	4	32
Temperature-Humidity Sensor DHT11	2	8
Rain Sensor Module	1	2
PIR Sensor Module HC-SR501	1	3
Magnetic Reed Switch	1	4
Breadboard Mini	4	6
Led & Buzzers	2	2
ΣΥΝΟΛΟ		157

Πίν. 6. Κόστος εξοπλισμού

4.3. Προετοιμασία του Raspberry

Η υλοποίηση του σεναρίου ξεκινάει με την προετοιμασία του Raspberry εγκαθιστώντας το λειτουργικό σύστημα. Στη συνέχεια γίνεται η ενεργοποίηση της απομακρυσμένης πρόσβασης καθώς και η εγκατάσταση των πακέτων docker και docker-compose. Για λειτουργικό σύστημα επιλέχθηκε Raspberry Pi OS Lite που δεν υποστηρίζει γραφικό περιβάλλον αλλά περιέχει όλα τα απαραίτητα πακέτα για την λειτουργία του raspberry. Η τελευταία έκδοση του Raspberry Pi OS έγινε διαθέσιμη τον Ιανουάριο του 2022. Για την προετοιμασία του Raspberry θα χρειαστεί ένας υπολογιστής με εγκατεστημένη την ansible, το git καθώς και ένα ζεύγος SSH κλειδιών.

4.3.1. Εγκατάσταση λειτουργικού συστήματος

Ένας τρόπος να γίνει η εγκατάσταση του λειτουργικού συστήματος είναι χρησιμοποιώντας το Raspberry Pi Imager που έχει γραφικό περιβάλλον ενώ ένας δεύτερος είναι από terminal με την εντολή dd. Τοποθετούμε την SD κάρτα στον υπολογιστή μας και τρέχουμε τις παρακάτω εντολές. Με την πρώτη εντολή φαίνεται ο αριθμός του δίσκου που έχει πάρει η κάρτα SD. Στη συνέχεια κάνουμε unmount και με την εντολή dd γράφουμε το image που έχουμε κατεβάσει από τον επίσημο ιστότοπο (https://www.raspberrypi.com/software/operating-systems/) στην SD κάρτα.

```
diskutil list
diskutil unmountDisk /dev/disk4
dd if=2022-01-28-raspios-bullseye-armhf-lite.img of=/dev/rdisk4 bs=1m
```

4.3.2. Ενεργοποίηση SSH

By default το ssh είναι απενεργοποιημένο στο raspberry. Το ενεργοποιούμε δημιουργώντας ένα κενό αρχείο με όνομα `ssh` στο root folder.

```
touch /Volumes/boot/ssh
```

Έπειτα κάνουμε unmount την SD κάρτα και την τοποθετούμε στο raspberry.

4.3.3. Εγκατάσταση του SSH Key

Αφού εκκινήσει το raspberry τρέχουμε την παρακάτω εντολή.

```
ssh-copy-id pi@raspberrypi
```

Έτσι θα μπορούμε να κάνουμε login στο raspberry χωρίς να εισάγουμε κωδικό πρόσβασης αλλά με το ιδιωτικό μας κλειδί (SSH key pair). Θα μας ζητήσει το default password που είναι το *raspberry*. Αν δεν υπάρχει local dns στην υποδομή μας αντί για raspberrypi βάζουμε την IP διεύθυνση που πήρε το raspberry.

4.3.4. Ορισμός Τοποθεσίας

Κάνουμε login στο raspberry και τρέχουμε την παρακάτω εντολή ορίζοντας έτσι την τοποθεσία μας ώστε να έχει το raspberry την σωστή ώρα.

```
ssh pi@raspberrypi
sudo timedatectl set-timezone Europe/Athens
```

4.3.5. Ενημέρωση Μεταβλητών

Στον υπολογιστή μας κατεβάζουμε το project από το git και συμπληρώνουμε το αρχείο `config.yml` με μεταβλητές της επιλογής μας. Οι μεταβλητές αναφέρονται στο όνομα και credentials για την βάση καθώς και credentials για την διαχείριση της influxdb και του grafana.

```
git clone https://github.com/theohitman/pms-thesis.git
cd pms-thesis
cp config.yml.example config.yml
```

4.3.6. Εγκατάσταση Docker και Docker-Compose

Για την υποστήριξη των containers απαιτείται η εγκατάσταση του docker και docker-compose στο raspberry. Από υπολογιστή που έχουμε εγκατεστημένη την ansible εκτελούμε το παρακάτω playbook όπου κάνει τις εξής ενέργειες στο raspberry:

```
ansible-playbook playbooks/docker-install.yml
```

1. Απενεργοποίηση του password authentication. SSH μόνο με κλειδί (SSH key pair)
2. Αναβάθμιση όλων των πακέτων του raspberry
3. Εγκατάσταση απαραίτητων πακέτων για το docker
4. Εγκατάσταση docker
5. Προσθήκη δικαιωμάτων στον χρήστη pi ώστε να τρέχει εντολές για το docker
6. Εγκατάσταση docker-compose

Τα playbooks που χρησιμοποιούνται στο σενάριο περιγράφονται αναλυτικά στο κεφάλαιο 3.3.

4.4. Διαμόρφωση μικροελεγκτών WeMos D1 mini ESP8266

Η συνέχεια της υλοποίησης περιλαμβάνει την διασύνδεση των επιμέρους components (αισθητήρων, leds, κτλ.) με τους μικροελεγκτές WeMos D1 mini ESP8266 καθώς και τον προγραμματισμό τους σύμφωνα με το σενάριο χρήσης. Τα σχεδιαγράμματα με τις διασυνδέσεις υπάρχουν με καλύτερη ανάλυση στον σύνδεσμο GitHub (https://github.com/theohitman/pms-thesis/tree/main/images) ενώ ο κώδικας όλων των μικροελεγκτών στον σύνδεσμο GitHub (https://github.com/theohitman/pms-thesis/tree/main/sketches).

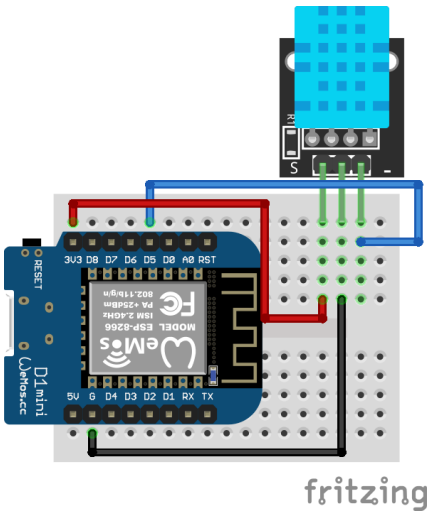
4.4.1. Προετοιμασία περιβάλλοντος Arduino IDE

Για να φορτώσουμε τον κώδικα στους μικροελεγκτές πρέπει να εγκαταστήσουμε και να παραμετροποιήσουμε το Arduino IDE ως εξής:

1. Κατεβάζουμε την εφαρμογή από το επίσημο site (https://www.arduino.cc/en/software) για το λειτουργικό που διαθέτουμε και την εγκαθιστούμε.
2. Από τα Preferences του Arduino IDE κάνουμε επικόλληση το παρακάτω URL στο πεδίο Additional Board Manager URLs. Αυτό γίνεται για να μπορέσει το Arduino IDE να διαβάσει και να συνδεθεί με το ESP8266.
   ```
   http://arduino.esp8266.com/stable/package_esp8266com_index.json
   ```
3. Από Tools > Board > Board Manager κάνουμε αναζήτηση το esp8266 επιλέγοντας την τελευταία έκδοση και επιλέγουμε Install.
4. Συνδέουμε τον μικροελεγκτή στον υπολογιστή και επιλέγουμε την σωστή πόρτα από τα Tools > Port
5. Έπειτα από Tools > Board > Επιλέγουμε το board που έχουμε συνδέσει. Στην περίπτωσή μας είναι το: LOLIN(WEMOS) D1 R2 & mini
6. Σύμφωνα με το sketch που θέλουμε να φορτώσουμε απαιτούνται και οι ανάλογες βιβλιοθήκες. Οι βιβλιοθήκες αυτές αν δεν υπάρχουν ήδη θα πρέπει να γίνουν import στο Arduino IDE. Από τα Tools > Manage Libraries... > Αναζητούμε την βιβλιοθήκη και επιλέγουμε Install. Θα πρέπει να εγκατασταθούν οι βιβλιοθήκες `DHT sensor library` και `PubSubClient`.
7. Τέλος για να φορτώσουμε τον κώδικα στον μικροελεγκτή αντιγράφουμε το sketch που θέλουμε σε ένα νέο παράθυρο του Arduino IDE, επιλέγουμε Verify για να το κάνει compile και Upload για να το φορτώσει στο μικροελεγκτή.

4.4.2. Συνδεσμολογία 1ου Μικροελεγκτή

Ο πρώτος μικροελεγκτής διασυνδέεται με έναν αισθητήρα DHT11 όπως φαίνεται στο παρακάτω σχήμα. Ο κώδικας για τον προγραμματισμό του βρίσκεται στο Παράρτημα Α'. Το τελικό αποτέλεσμα φαίνεται στη παρακάτω φωτογραφία.

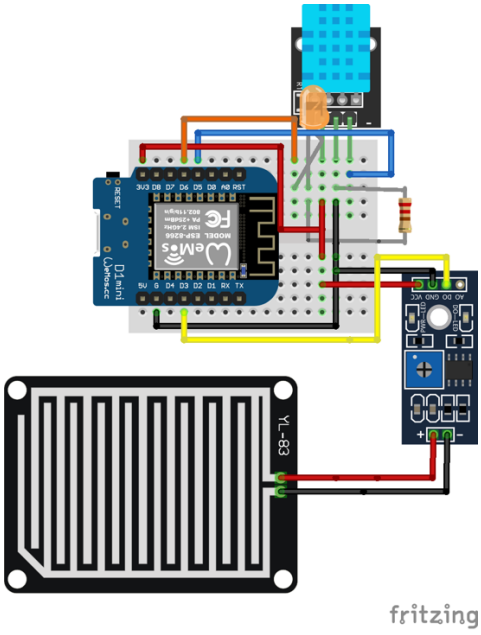


Σχ. 5. ESP1

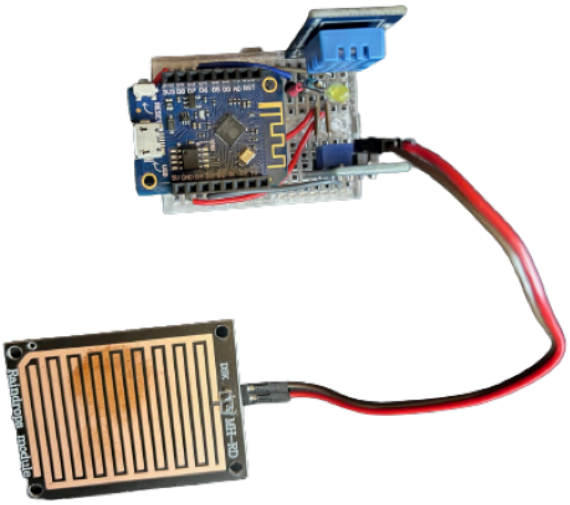
Εικ. 27. ESP1

4.4.3. Συνδεσμολογία 2ου Μικροελεγκτή

Ο δεύτερος μικροελεγκτής εκτός από τον αισθητήρα θερμοκρασίας υγρασίας DHT11 έχει συνδεδεμένο και έναν αισθητήρα νερού. Επίσης προστέθηκε ένα led που ανάβει σε περίπτωση βραχυκύκλωσης του διακόπτη στον αισθητήρα νερού. Με αυτόν τον τρόπο θα μπορεί να ελεγχθεί η ορθή λειτουργία του.



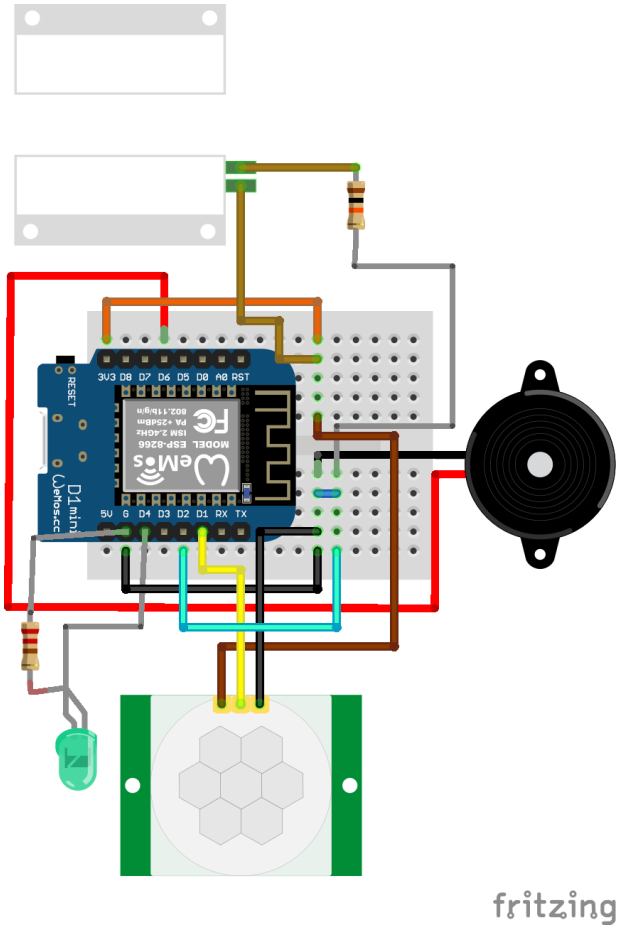
Σχ. 6. ESP2



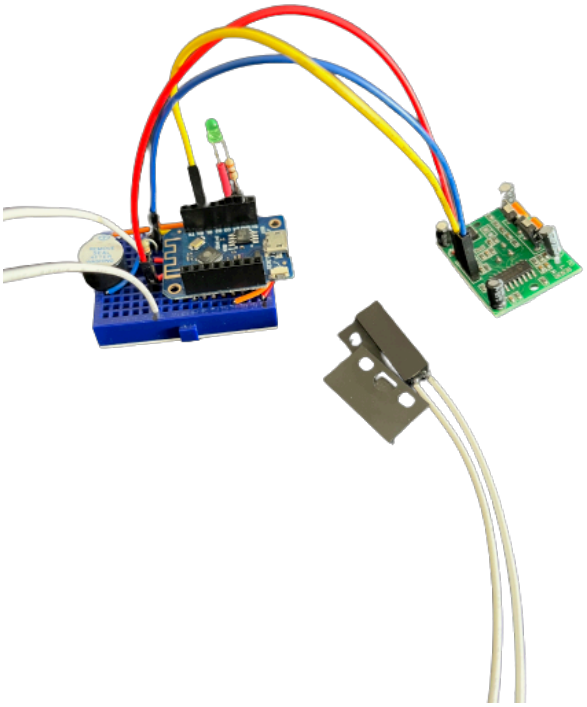
Εικ. 28. ESP2

4.4.4. Συνδεσμολογία 3ου Μικροελεγκτή

Ο τρίτος μικροελεγκτής έχει μια πιο πολύπλοκη συνδεσμολογία όπως φαίνεται και στο παρακάτω σχήμα, αφού διασυνδέεται με έναν αισθητήρα ανίχνευσης κίνησης και έναν μαγνητικό διακόπτη. Επιπλέον έχει προστεθεί ένα buzzer το οποίο ενεργοποιείται όταν η πόρτα του data center ανοίγει, έτσι ώστε να ειδοποιείται το άτομο που θα εισέρχεται στον χώρο και να μην χάνεται η ψύξη. Ένα led επίσης ενεργοποιείται όταν ο ανιχνευτής κίνησης εντοπίσει κίνηση έτσι ώστε να υπάρχει έλεγχος για την ορθή λειτουργία του.



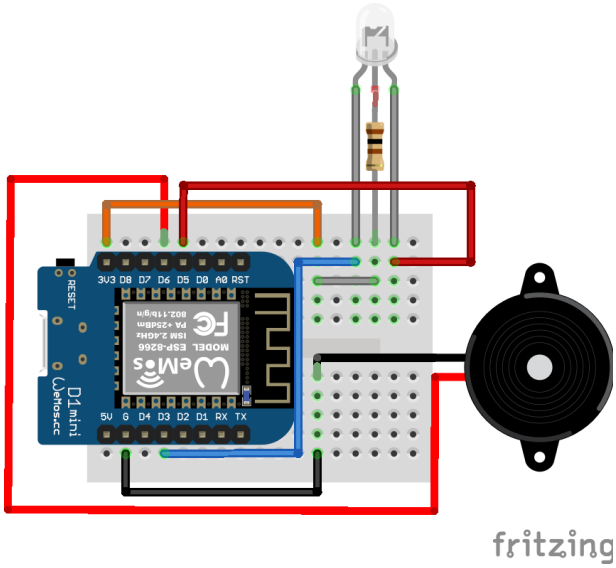
Σχ. 7. ESP3



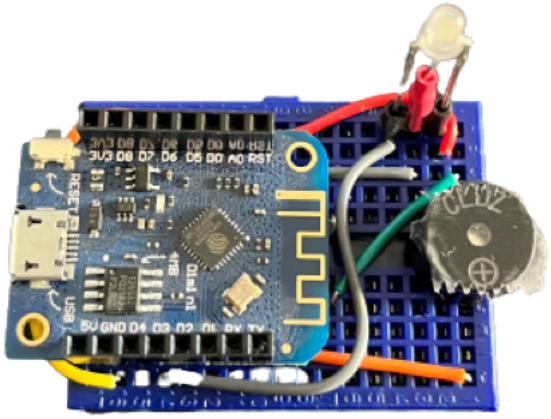
Εικ. 29. ESP3

4.4.5. Συνδεσμολογία 4ου Μικροελεγκτή

Αυτός ο μικροελεγκτής διασυνδέεται με ένα buzzer και ένα led το οποίο έχει την δυνατότητα να ανάψει σε μπλε ή κόκκινο χρώμα.



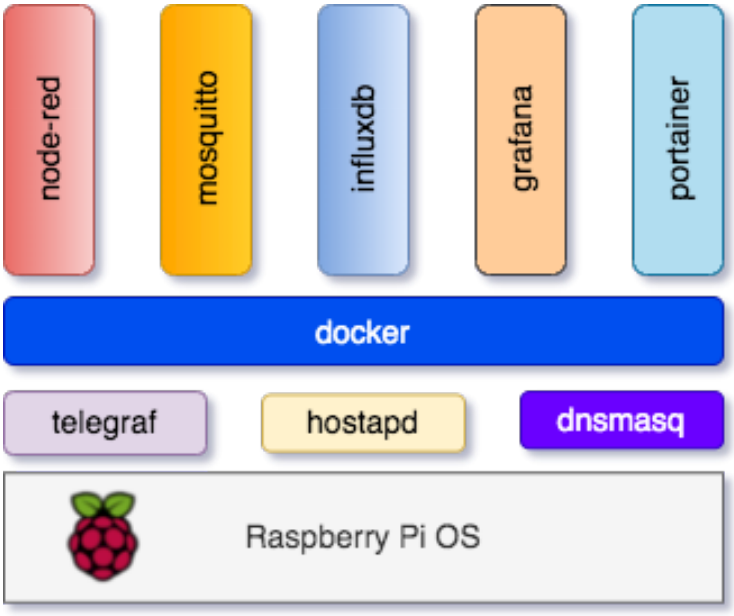
Σχ. 8. NOC



Εικ. 30. NOC

4.5. Stack Deployment

Σε αυτό το στάδιο της υλοποίησης θα γίνει η εγκατάσταση του δικτύου αισθητήρων, η εγκατάσταση και παραμετροποίηση του Telegraf και η ανάπτυξη όλων των containers με τις εφαρμογές που χρειάζεται το σύστημα όπως φαίνεται στο παρακάτω σχήμα.



Σχ. 9. Stack Deployment

4.5.1. Εγκατάσταση δικτύου αισθητήρων

Όπως αναφέρθηκε και στο κεφάλαιο 3.1 θα χρησιμοποιηθεί η ασύρματη κάρτα δικτύου του raspberry για την υλοποίηση ενός δικτύου αισθητήρων. Σε αυτό το δίκτυο θα συνδέονται μόνο οι μικροελεγκτές ESP8266. Με το παρακάτω playbook γίνονται οι εξής ενέργειες στο raspberry:

1. Εγκατάσταση πακέτων hostapd και dnsmasq
2. Εκκίνηση της ασύρματης κάρτας δικτύου και ορισμός IP διεύθυνσης σε αυτή
3. Ρυθμίσεις για DHCP server (dnsmasq) για τον διαμοιρασμό IP διευθύνσεων
4. Ρυθμίσεις για access point (hostapd) που θα συνδέονται οι μικροελεγκτές

```
ansible-playbook playbooks/sensor-network.yml
```

4.5.2. Εγκατάσταση Telegraf

Με αυτό το playbook εγκαθίσταται στο raspberry το telegraf καθώς και φορτώνεται το configuration για την βάση που θα αποθηκεύονται οι μετρήσεις. Σκοπός του telegraf agent είναι η παρακολούθηση της υγείας του raspberry pi.

```
ansible-playbook playbooks/telegraf-install.yml
```

4.5.3. Ανάπτυξη του Stack

Το τελευταίο playbook που θα χρειαστεί να τρέξουμε αναπτύσσει όλα τα containers με τις εφαρμογές που χρειάζεται το σύστημα. Τα εν λόγω containers φαίνονται στο Σχήμα 8 ενώ για το container του node-red γίνεται build ώστε να έρθει με εγκατεστημένα τα παρακάτω nodes.

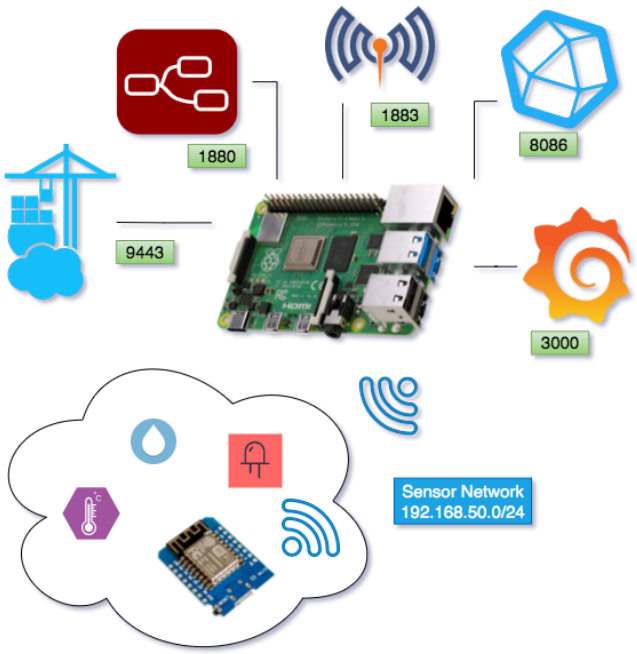
- node-red-contrib-influxdb
- node-red-dashboard
- node-red-node-email
- node-red-node-pi-gpio
- node-red-node-twilio

```
ansible-playbook playbooks/deploy-stack.yml
```

Με την εντολή ***docker ps*** και αφού συνδεθούμε με ssh στο rarsberry μπορούμε να ελέγξουμε ότι είναι up τα containers. Τα παραπάνω playbooks περιγράφονται αναλυτικά στο κεφάλαιο 3.3.

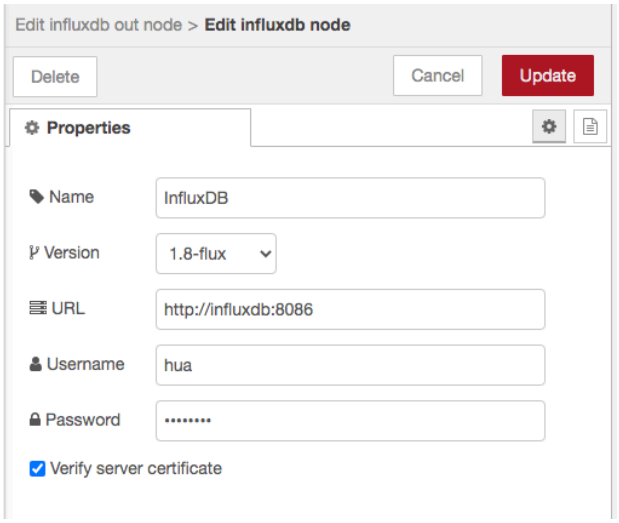
4.6. Τελικές Ρυθμίσεις

Το stack έχει πλέον σηκωθεί και απομένουν κάποιες τελευταίες ρυθμίσεις στα επιμέρους components του project. Οι πόρτες που ακούνε οι εφαρμογές φαίνονται στο παρακάτω σχέδιο.



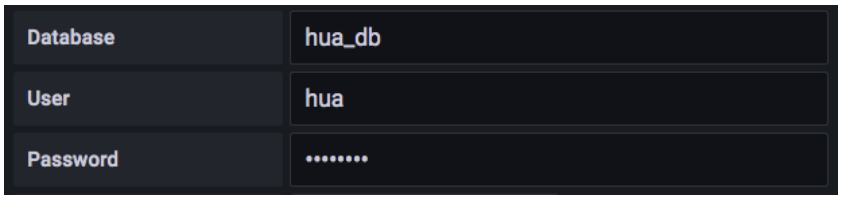
Σχ. 10. Port Mapping

Στο Node-RED στην καρτέλα InfluxDB επιλέγουμε ένα οποιοδήποτε InfluxDB node και ρυθμίζουμε την database, το username και password που ορίσαμε στο αρχείο config.yml. Επίσης στο Node-RED στην καρτέλα Alarms θα πρέπει να ορίσουμε email που θέλουμε να έρχονται οι ειδοποιήσεις και κινητό τηλέφωνο στο twillio node αν επιθυμούμε ειδοποιήσεις μέσω sms. Για την λειτουργία αυτή θα πρέπει να έχει προηγηθεί εγγραφή στο αντίστοιχο site.



Εικ. 31. Configure InfluxDB on node-red

Αντίστοιχη ρύθμιση με το node-red θα πρέπει να γίνει και στο Grafana ώστε να διαβάζει τις μετρήσεις από την βάση. Από Configuration > Data sources > Add data source. Εδώ επιλέγουμε influxdb και συμπληρώνουμε τα πεδία URL με http://influxdb:8086 και Database, User και Password ίδια με αυτά που συμπληρώσαμε στο αρχείο config.yml.



Εικ. 32. Configure InfluxDB on Grafana

Για να εισάγουμε το Dashboard που έχει δημιουργηθεί αποκλειστικά για αυτό το σενάριο επιλέγουμε Import και Upload JSON file. Στον φάκελο dashboards υπάρχουν τα διαθέσιμα dashboards και επιλέγουμε το Data Center.json. Στην παρακάτω εικόνα φαίνεται το τελικό αποτέλεσμα στο Grafana.

Εικ. 33. Data Center Dashboard

Για περισσότερες πληροφορίες της υγείας και των λειτουργιών του raspberry μπορούμε να εισάγουμε το Dashboard που δημιούργησε ο Jorge de la Cruz επιλέγοντας import > Import via grafana.com > 10578 > Load [21].

ΚΕΦ.5: ΣΥΖΗΤΗΣΗ - ΣΥΜΠΕΡΑΣΜΑΤΑ

Σε αυτό το κεφάλαιο θα αναφερθούν οι δυσκολίες που προέκυψαν κατά την υλοποίηση της εργασίας και ο τρόπος που αυτές ξεπεράστηκαν. Τέλος θα αναφερθούν μελλοντικές βελτιώσεις που θα μπορούσαν να εφαρμοστούν καθώς και τα τελικά συμπεράσματα.

Η επιλογή του κατάλληλου εξοπλισμού για την υλοποίηση του συστήματος ήταν ένα από τα αρχικά προβλήματα που παρουσιάστηκαν. Το πλήθος των διαθέσιμων IoT development boards αλλά και διαφορετικών αισθητήρων που κυκλοφορούν διαθέσιμα στην αγορά έκανε την επιλογή ακόμα πιο δύσκολη. Τα διαφορετικά DIY projects που κυκλοφορούν στα διαδίκτυο καθώς και η εμπειρία του φίλου και συναδέλφου Κου Γιώργου Καρούμπαλη σε αυτόν τον τομέα με κατεύθυναν στην επιλογή των παραπάνω υλικών με αποτέλεσμα να ξεπεραστεί αυτό το πρώτο εμπόδιο.

Ο προγραμματισμός των μικροελεγκτών ESP8266 ήταν ένα ακόμα εμπόδιο που με δυσκόλεψε. Το περιβάλλον του Arduino IDE καθώς και ο κώδικας που απαιτείται για την παραμετροποίηση του εκάστοτε αισθητήρα που θα συνδεθεί με τον μικροελεγκτή ήταν κάτι καινούριο για εμένα. Το ESP8266 είναι μια διαδεδομένη πλατφόρμα που χρησιμοποιείται αρκετά χρόνια σε υλοποιήσεις IoT και δεν επιλέχθηκε τυχαία. Στο διαδίκτυο υπάρχει διαθέσιμο υλικό με παραδείγματα και οδηγούς για παρεμφερείς υλοποιήσεις. Αυτό που έπρεπε να κάνω ήταν να κατανοήσω τον κώδικα και να τον προσαρμόσω κατάλληλα ώστε να καλύπτει τις ανάγκες του δικού μου project.

Η διαδικασία της εγκατάστασης αλλά και επανεγκατάστασης του συστήματος ήθελα να γίνει με όσο το δυνατόν πιο γρήγορο και αυτοματοποιημένο τρόπο αποκρύπτοντας από τον χρήστη ενδιάμεσα βήματα όπως είναι η εγκατάσταση και αναβάθμιση πακέτων, η εγκατάσταση του δικτύου αισθητήρων, ρυθμίσεις των επιμέρους components κ.α. Για το λόγο αυτό έπρεπε να θυμηθώ και να εμβαθύνω τις γνώσεις μου στην Ansible. Η αναδρομή στις σημειώσεις του μαθήματος Προηγμένες Τεχνικές Διαχείρισης Υπολογιστικών Υποδομών και το βιβλίο του Jeff Geerling - Ansible for DevOps που είχε προτείνει ο Κος Τσαδήμας με βοήθησε να ξεπεράσω και αυτό το εμπόδιο [22].

Μια ακόμα τεχνολογία που έπρεπε να χρησιμοποιήσω για την υποστήριξη των επιμέρους υπηρεσιών του συστήματος ήταν το Docker και το Docker Compose. Η πλειοψηφία των υπηρεσιών είχαν διαθέσιμο image στο Docker Hub που μπορούσα να εκμεταλλευτώ. Στην περίπτωση του Node-RED όμως και για να έχω διαθέσιμα τα απαιτούμενα nodes έπρεπε να κάνω build ένα custom image. Και σε αυτή την περίπτωση έπρεπε να ανατρέξω σε σημειώσεις και προτεινόμενη βιβλιογραφία του μαθήματος Διαχείριση Δικτυακών και Υπολογιστικών Πόρων από τους κυρίους Δημόπουλο και Τσαδήμα.

Ένα σημαντικό δίλημμα που έπρεπε να λύσω ήταν ο τρόπος που θα γράφονται στη βάση οι μετρήσεις από τους διάφορους αισθητήρες. Κάποιες από τις επιλογές μου θα μπορούσαν να ήταν μέσω του Telegraf ή μέσω ενός custom python script που θα δημιουργούσα. Και οι δύο αυτές λύσεις θα έκαναν την δουλειά αλλά θα ήταν δύσκολη και χρονοβόρα η παραμετροποίηση του συστήματος σε περίπτωση οποιασδήποτε αλλαγής στο πλήθος των αισθητήρων, στην αλλαγή ονόματος της βάσης ή του MQTT topic κ.α. Λύση σε αυτό το πρόβλημα έδωσε το Node-RED όπου με γραφικό περιβάλλον και ελάχιστο κώδικα μπορεί εύκολα να υποστηρίξει τέτοιου είδους παραμετροποιήσεις.

Σκοπός της εργασίας ήταν η ανάπτυξη ενός αυτόνομου συστήματος παρακολούθησης περιβαλλοντολογικών συνθηκών σε ένα κέντρο δεδομένων με όσο το δυνατόν απλούστερο υλικό και λογισμικό ανοιχτού κώδικα. Η χρήση του Raspberry Pi και των μικροελεγκτών ESP8266 προγραμματιζόμενα με το Arduino IDE συνετέλεσαν στην επίτευξη του παραπάνω σκοπού. Επιπροσθέτως η ανάπτυξη του συστήματος θα έπρεπε να ικανοποιεί συνθήκες εύκολης εγκατάστασης, παραμετροποίησης αλλά και επέκτασης με περισσότερους αισθητήρες. Η χρήση εργαλείων και τεχνολογιών όπως η ansible, το docker, το MQTT πρωτόκολλο επικοινωνίας και το node-red βοήθησαν στην ικανοποίηση όλων αυτών των συνθηκών. Το χαμηλό κόστος των υλικών που απαιτούνται και η ευκολία υλοποίησης κάνει το σύστημα ελκυστικό ακόμα και για οικιακή χρήση.

Στο υπάρχον σύστημα θα μπορούσαν σίγουρα να υπάρχουν προσθήκες και βελτιώσεις. Η ανάπτυξη μιας εφαρμογής που θα παρήγαγε τον απαιτούμενο κώδικα για ένα Sketch θα ήταν μια πολύ καλή βελτίωση του συστήματος. Η εφαρμογή θα μπορούσε να παίρνει σαν είσοδο το

πλήθος και το είδος των αισθητήρων που απαιτείται να υποστηριχθεί από έναν μικροελεγκτή και να εξάγει τον κώδικα αλλά και την προτεινόμενη συνδεσμολογία για αυτόν.

Σε ένα απαιτητικό και ευαίσθητο περιβάλλον χρήσης του συστήματος που αναπτύχθηκε, η διακοπή των προσφερόμενων υπηρεσιών θα ήταν ένα σοβαρό πρόβλημα. Για την ελαχιστοποίηση αυτής της πιθανότητας και την αύξηση της διαθεσιμότητας του συστήματος μια υλοποίηση υψηλής διαθεσιμότητας θα ήταν μια πολύ καλή βελτίωση. Το Docker Swarm HA και το Kubernetes HA cluster είναι δύο υλοποιήσεις που θα μπορούσαν να προσφέρουν υψηλή διαθεσιμότητα στο σύστημα [23].

Βελτιώσεις από πλευράς ασφάλειας θα μπορούσαν να γίνουν σε διάφορα components του συστήματος. Η ενεργοποίηση του HTTPS πρωτοκόλλου για πρόσβαση στο Node-RED [24], η αυθεντικοποίηση των clients του MQTT broker με κωδικούς πρόσβασης ή με πιστοποιητικά SSL θα ήταν μερικές από αυτές [25].

Συμπερασματικά λοιπόν καταλήγουμε ότι χρησιμοποιώντας τεχνολογίες και λογισμικό ανοιχτού κώδικα που είναι διαθέσιμα για όλους μπορούμε να πετύχουμε την ανάπτυξη συστημάτων που καλύπτουν τις ανάγκες για ειδοποίηση, παρακολούθηση και διαχείριση της θερμοκρασίας και υγρασίας για ένα μικρό ή μεγαλύτερο κέντρο δεδομένων.

Το πεδίο χρήσης του συστήματος δεν περιορίζεται στα Data Centers αλλά θα μπορούσε να χρησιμοποιηθεί σε θερμοκήπια, βιομηχανικές μονάδες, ψυκτικούς θαλάμους και γενικά σε χώρους όπου η περιβαλλοντολογικές συνθήκες είναι απαραίτητο να παρακολουθούνται και να καταγράφονται.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] L. A. Barroso, U. Holzle και P. Ranganathan, The Datacenter as a Computer: Designing Warehouse-Scale Machines, Third Edition, San Rafael, California: Morgan & Claypool, 2018.

[2] What are the Differences Between Raspberry Pi and Arduino?, https://www.electronicshub.org/raspberry-pi-vs-arduino/. [Πρόσβαση Φεβρουάριος 2022].

[3] Raspberry Pi, https://en.wikipedia.org/wiki/Raspberry_Pi.[Πρόσβαση Φεβρουάριος 2022].

[4] A Complete Guide to Microcontrollers for IoT, https://www.nabto.com/iot-microcontroller-guide/. [Πρόσβαση Φεβρουάριος 2022].

[5] Top 25 IoT Development Boards In 2022 And How To Choose The Right One, https://www.intuz.com/guide-on-top-iot-development-boards. [Πρόσβαση Φεβρουάριος 2022].

[6] ESP8266, https://en.wikipedia.org/wiki/ESP8266. [Πρόσβαση Φεβρουάριος 2022].

[7] ESP8266 for IoT: A Complete Guide, https://www.nabto.com/esp8266-for-iot-complete-guide/. [Πρόσβαση Φεβρουάριος 2022].

[8] What Is a Breadboard and How Does It Work?, https://www.makeuseof.com/tag/what-is-breadboard/. [Πρόσβαση Φεβρουάριος 2022].

[9] ESPEasy, https://www.letscontrolit.com/wiki/index.php/ESPEasy. [Πρόσβαση Φεβρουάριος 2022].

[10] Sonoff wifi switch control with ESP Easy and Node Red, https://www.youtube.com/watch?v=8CuD6YBdACs. [Πρόσβαση Φεβρουάριος 2022].

[11] Raspberry Pi OS, https://en.wikipedia.org/wiki/Raspberry_Pi_OS. [Πρόσβαση Φεβρουάριος 2022].

[12] Everything you need to know about Arduino Code, https://www.circuito.io/blog/arduino-code/. [Πρόσβαση Μάρτιος 2022].

[13] R. Light, «Mosquitto: server and client implementation of the MQTT protocol,» The Journal of Open Source Software, 2017.

[14] MQTT, https://mqtt.org/. [Πρόσβαση Φεβρουάριος 2022].

[15] N. Poulton, Docker Deep Dive, 2018.

[16] Raspberry Pi, - Hotspot/Access Point dhcpcd method https://www.raspberryconnect.com/projects/65-raspberrypi-hotspot-accesspoints/168-raspberry-pi-hotspot-access-point-dhcpcd-method. [Πρόσβαση Φεβρουάριος 2022].

[17] How to use your Raspberry Pi as a wireless access point, https://thepi.io/how-to-use-your-raspberry-pi-as-a-wireless-access-point/. [Πρόσβαση Φεβρουάριος 2022].

[18] R. Santos, *Build a Home Automation System for $100,* Porto: RNTLab.com.

[19] R. Santos και S. Santos, *ESP8266 Web Server with Arduino IDE,* Porto: RNTLab.com.

[20] Andrei Ochmat, https://discourse.nodered.org/u/andrei/summary. [Πρόσβαση Φεβρουάριος 2022].

[21] Raspberry Pi Monitoring, https://grafana.com/grafana/dashboards/10578. [Πρόσβαση Μάρτιος 2022].

[22] J. Geerling, Ansible for DevOps, Leanpub, 2020.

[23] Implementing High Availability with Docker Swarm, https://dockerlabs.collabnix.com/intermediate/Implementing_High_Availability_with_Docker_Swarm.html. [Πρόσβαση Απρίλιος 2022].

[24] Securing Node-RED, https://nodered.org/docs/user-guide/runtime/securing-node-red. [Πρόσβαση Απρίλιος 2022].

[25] How to Install and Secure the Mosquitto MQTT Messaging Broker on Debian 10, https://www.digitalocean.com/community/tutorials/how-to-install-and-secure-the-mosquitto-mqtt-messaging-broker-on-debian-10. [Πρόσβαση Απρίλιος 2022].

[26] J. Nickoloff και S. Kuenzli, Docker In Action, Shelter Island: Manning, 2019.

[27] Ε. Δουμάνης, *Σχεδίαση και Υλοποίηση Ολοκληρωμένου Συστήματος Ελέγχου Data Center με Χρήση Πλατφόρμας Arduino,* Λαμία: Πανεπιστήμιο Θεσσαλίας, 2020.

[29] Α. Παπαευσταθίου, *Σύστημα παρακολούθησης & ειδοποίησης συμβάντων σε datacenter, με χρήση IOT,* Αθήνα: Χαροκόπειο Πανεπιστήμιο, 2020.

[30] Ν. Μ. Βαλάσης, *Σύστημα παρακολούθησης και περιορισμού ενεργειακής κατανάλωσης κτιρίου,* Αθήνα: Χαροκόπειο Πανεπιστήμιο, 2019.