Σχολή Θετικών Επιστημών και Τεχνολογίας

Μεταπτυχιακή Εξειδίκευση στα Πληροφοριακά Συστήματα

Διπλωματική Εργασία

Έξυπνο Σύστημα Ελέγχου Απόδοσης Κεντρικής Θέρμανσης Κτηρίων

Βενιζέλος - Παναγιώτης Παπανικολάου

Επιβλέπων καθηγητής: Θεοφάνης Ορφανουδάκης

Πάτρα, Σεπτέμβριος 2021

Έξυπνο Σύστημα Ελέγχου Απόδοσης Κεντρικής Θέρμανσης Κτηρίων

Βενιζέλος - Παναγιώτης Παπανικολάου

Επιτροπή Επίβλεψης Διπλωματικής Εργασίας

Επιβλέπων Καθηγητής:	Συν-Επιβλέπων Καθηγητής:
Θεοφάνης Ορφανουδάκης	Κωνσταντίνος Χαϊκάλης
Αναπληρωτής Καθηγητής	Επίκουρος Καθηγητής
Ελληνικό Ανοικτό Πανεπιστήμιο	Πανεπιστήμιο Θεσσαλίας

Πάτρα, Σεπτέμβριος 2021

Ευχαριστώ τους καθηγητές κ. Θεοφάνη Ορφανουδάκη και κ. Κωνσταντίνο Χαϊκάλη που μου ανάθεσαν την παρούσα διπλωματική εργασία και μου επέτρεψαν να την υλοποιήσω.

Περίληψη

Τα έξυπνα και αυτόματα συστήματα θέρμανσης εκμεταλλεύονται την ικανότητα των τεχνολογιών του διαδικτύου των πραγμάτων (Internet of Things) βελτιώνοντας την ποιότητα και τις επιδόσεις τους. Η συλλογή και η ανάλυση των δεδομένων διαμέσου των αισθητήρων του δικτύου κάνει δυνατή την πρόβλεψη της ζήτησης για την παραγωγή θέρμανσης, δημιουργώντας ένα πραγματικά ενεργειακά αποδοτικό σύστημα.

Στην παρούσα εργασία δημιουργείται ένα έξυπνο σύστημα ελέγχου και απόδοσης της κεντρικής θέρμανσης κτηρίου. Η συγκεκριμένη υλοποίηση χρησιμοποιώντας μικροελεγκτές, απαλλάσσει την λειτουργία του θερμοστάτη από την ανάγκη καλωδίωσης για την τροφοδοσία του και προσφέρει την δυνατότητα μετατροπής υπαρχόντων κτηρίων σε έξυπνα. Επίσης επιδεικνύεται η διαδικασία αποθήκευσης, ανάλυσης και οπτικοποίησης δεδομένων από δωρεάν εργαλεία (open source) τα οποία μπορούν να χρησιμοποιηθούν από τον οποιονδήποτε ανεξάρτητα από το επίπεδο προγραμματισμού που διαθέτει.

Το σύστημα που υλοποιήθηκε χωρίζεται σε τέσσερα υποσυστήματα. Το υποσύστημα του εσωτερικού χώρου αποτελείται από τέσσερις μικροελεγκτές ESP32, που επικοινωνούν μεταξύ τους διαμέσου του πρωτοκόλλου Bluetooth Low Energy προκειμένου να μεταδίδουν τις μετρήσεις των αισθητήρων που κατέχουν. Το υποσύστημα του εξωτερικού χώρου και το υποσύστημα του λεβητοστασίου αποτελούνται από έναν μικροελεκτή ESP32 που με την χρήση αισθητήρων συλλέγουν δεδομένα από το περιβάλλον γύρο τους. Τα προαναφερθέντα υποσυστήματα μεταδίδουν τα δεδομένα που συλλέχθηκαν διαμέσου του πρωτοκόλλου επικοινωνίας MQTT στον διακομιστή, όπου αργότερα θα γίνει η επεξεργασία τους. Το υποσύστημα αποθήκευσης και ανάλυσης δεδομένων αποτελείται από ένα Raspberry Pi 4, το οποίο λειτουργεί σαν διακομιστής. Φιλοξενεί τα δωρεάν προς χρήση λογισμικά Mosquitto broker, Node-Red, InfluxDB και Grafana, η χρήση των οποίον κάνει δυνατή την διαδικασία αποθήκευσης, οπτικοκοποίσης και ανάλυσης των δεδομένων.

Τέλος παρουσιάζονται τα γραφήματα που αναπτύχθηκαν με την χρήση των εφαρμογών Node-Red και Grafana, αποδεικνύοντας πως από δωρεάν εργαλεία μπορούν να δημιουργηθούν επαγγελματικές εφαρμογές. Δίνεται ιδιαίτερη έμφαση στην ικανότητα οπτικοποίησης των δεδομένων διαμέσου της εφαρμογής Grafana, προσφέροντας ιδανικές λύσεις ανάλυσης προς τον χρήστη.

Λέξεις – Κλειδιά

Διαδίκτυο Των Πραγμάτων, Έξυπνος Θερμοστάτης, Έξυπνα Κτήρια, MQTT, ESP32, Bluetooth Low Energy (BLE), Σύστημα Απόκτησης Δεδομένων, Παρακολούθηση Σε Πραγματικό Χρόνο, Raspberry Pi, Node-Red, Grafana, Σύστημα Ελέγχου Αισθητήρων

Central Heating Smart Metering Control System

Venizelos - Panagiotis Papanikolaou

Abstract

Smart and automated heating systems, take advantage of the ability offered by the Internet of Things, improving their quality and performance. The collection and analysis of data through the network's sensors makes it possible to predict the demand for heating, thus creating a truly efficient system.

In this dissertation, a smart system that controls and measures a building's central heating unit is created. This specific implementation using microcontrollers, relieves the thermostat's need for wiring in order to get energy supply and offers the possibility of converting existing buildings into smart ones. It also demonstrates the process of storing, analysing and visualising data using open-source (free) tools that can be used by anyone regardless of their level of expertise in programming.

The implemented system is divided into four subsystems. The indoor subsystem consists of four ESP32 microcontrollers, which communicate with each other via the Bluetooth Low Energy (BLE) protocol in order to transmit the sensors' measurements. The outdoor and the boiler-room subsystems, each consist of an ESP32 microcontroller that uses sensors to collect data from their surroundings. The aforementioned subsystems transmit the collected data via the MQTT communication protocol to the server, where it will later be processed.

The data storage and analysis subsystem consists of a Raspberry Pi 4, which works as a server. It hosts the open-source software Mosquitto broker, Node-Red, InfluxDB and Grafana the use of which, enables the data storage, analysis and visualisation processes.

Finally, the graphs which were developed using the Node-Red and Grafana applications are presented, proving that open-source software can be used to create professional applications. Specific emphasis is placed on the ability to create data visualisation using Grafana, offering ideal analytical solutions to the user.

Keywords

Internet Of Things, Smart Thermostat, Smart Buildings, MQTT, ESP32, Bluetooth Low Energy (BLE), Data Acquisition System, Real-Time Monitoring, Raspberry Pi, Node-Red, Grafana, Sensor Control System

Περιεχόμενα

Περίληψη v
Abstract vii
Περιεχόμενα ix
Κατάλογος Εικόνων / Σχημάτων xi
Συντομογραφίες & Ακρωνύμια xiii
1 Εισαγωγή 1
1.1 Πρόλογος 1
1.2 Σκοπός της εργασίας 1
1.3 Δομή της εργασίας 1
2 Έξυπνο Σύστημα Ελέγχου Απόδοσης Κεντρικής Θέρμανσης Κτηρίων 3
2.1 Το Internet of Things σαν σύστημα αυτοματισμού 3
2.2 Ο ρόλος του Internet Of Things στους οικιακούς αυτοματισμούς 4
2.3 Προδιαγραφές και σχεδιασμός συστήματος ελέγχου απόδοσης κεντρικής θέρμανσης 6
2.3.1 Απαιτήσεις Συστήματος 8
2.3.2 Σχεδιασμός συστήματος 9
3 Δίκτυα Και Πρωτόκολλα Επικοινωνίας 12
3.1 WIFI 12
3.2 Bluetooth Low Energy (BLE) 13
3.3 Πρωτόκολλο Επικοινωνίας MQTT 16
4 Παρουσίαση Συστήματος 21
4.1 ESP32 21
4.2 Αισθητήρες 24
4.2.1 Αισθητήρας Θερμοκρασίας Και Υγρασίας DHT11 25
4.2.2 Αισθητήρας Θερμοκρασίας DS18B20 26
4.2.3 Αισθητήρας Φωτεινότητας BH1750 26
4.3 Relay 27
4.4 Πρόσθετα Βοηθητικά Υλικά 28
4.5 Σύστημα 29
4.5.1 Υποσύστημα Εσωτερικού Χώρου 30
4.5.2 Υποσύστημα Εξωτερικού Χώρου 34
4.5.3 Υποσύστημα Λεβητοστασίου 35
5 Συλλογή Και Επεξεργασία Δεδομένων Συστήματος 36
5.1 Raspberry Pi 4 36
5.2 Raspberry Pi OS 38
5.3 Eclipse Mosquitto MQTT Broker 39
5.4 Node-Red 39
5.5 InfuxDB 43
5.6 Grafana 46
6 Παρουσίαση Εφαρμογών Διεπαφής Χρήστη 48
6.1 Παρουσίαση Εφαρμογής με Node-Red 48
6.1.1 Dashboard του Node-Red 48
6.1.2 Πρόγραμμα στο Node-Red 50

6.2 Παρουσίαση Εφαρμογής με Grafana 54
6.2.1 Dashboard του Grafana 54
6.2.2 Περιβάλλον Ρυθμίσεων Πίνακα και Εξαγωγή Αρχείου CSV 58
7 Αποτελέσματα Χρήσης Συστήματος 62
7.1 Αποτελέσματα 62
7.1.1 Αποτελέσματα Οπτικοποίησης και Ανάλυση Δεδομένων 62
7.1.2 Λειτουργικότητας συστήματος 66
7.2 Συμπεράσματα 71
7.2.1 Συμπεράσματα και προβλήματα κατά την υλοποίηση 72
7.3 Μελλοντική Έρευνα 73
Βιβλιογραφία 75
Παράρτημα Α: ΚΩΔΙΚΑΣ ΣΥΣΤΗΜΑΤΟΣ ΘΕΡΜΟΣΤΑΤΗ 78
Παράρτημα Β: ΕΓΚΑΤΑΣΤΑΣΗ ΛΟΓΙΣΜΙΚΩΝ ΔΙΑΚΟΜΙΣΤΗ RASPBERRY PI 4 118
Παράρτημα Γ: ΚΩΔΙΚΑΣ ΣΤΟ NODE-RED ΣΕ ΜΟΡΦΗ COMPACT JSON 136

Κατάλογος Εικόνων / Σχημάτων

Εικόνα 1 Σύστημα Κεντρικής Θέρμανσης Κτηρίου 6
Εικόνα 2 Ιεραρχική Δομή BLE Μηνυμάτων 15
Εικόνα 3 Αναπαράσταση της διάταξης και της λειτουργίας του MQTT 19
Εικόνα 4 Pinout του ESP32-WROOM-32 (Πηγή: esp32, https://esp32.com/) 22
Εικόνα 5 Chip ESP32-WROOM-32 (Πηγή: digikey, https://www.digikey.lv/en/products/detail/espressif-systems/ESP32-WROOM-32-8MB/9381728) 23
Εικόνα 6 Πλακέτες Lolin32, Doit 30pins και Doit 36pins αντίστοιχα 24
Εικόνα 7 Αισθητήρας DHT11 (Πηγή: components101, https://components101.com/sensors/dht11-temperature-sensor) 25
Εικόνα 8 Αισθητήρας DS18B20 26
Εικόνα 9 Αισθητήρας BH1750 (Πηγή: components101, https://components101.com/sensors/bh1750-ambient-light-sensor) 27
Εικόνα 10 Relay (Πηγή: components101, https://components101.com/switches/5v-four-channel-relay-module-pinout-features-applications-working-datasheet) 28
Εικόνα 11 Πρόσθετα Υλικά 29
Εικόνα 12 Τρόπος Επικοινωνίας Συστήματος 30
Εικόνα 13 Πλακέτες Εσωτερικού Χώρου 31
Εικόνα 14 Node 1 gateway 32
Εικόνα 15 Node 2 32
Εικόνα 16 Node 3 33
Εικόνα 17 Node 4 34
Εικόνα 18 Node 5 34
Εικόνα 19 Node 6 35
Εικόνα 20 Raspberry Pi 4 Συστήματος 36
Εικόνα 21 Ports Raspberry Pi 4 (Πηγή: raspberrypi, https://www.raspberrypi.org/products/raspberry-pi-4-model-b/) 37
Εικόνα 22 Node-Red Editor 41
Εικόνα 23 Node-Red Function 42
Εικόνα 24 Node-Red import/export 42
Εικόνα 25 TICK Stack 44
Εικόνα 26 Node-Red «ΚΕΝΤΡΙΚΗ ΣΕΛΙΔΑ» 49
Εικόνα 27 Node-Red Menu 50
Εικόνα 28 Node-Red IDE 51
Εικόνα 29 Node-Red Function Node 52
Εικόνα 30 Node-Red Client MQTT – Client InfluxDB 53
Εικόνα 31 Node-Red Additional Functions 53
Εικόνα 32 Node-Red topic time - relay 54
Εικόνα 33 Grafana Αρχική Σελίδα 55
Εικόνα 34 Grafana Boiler Room Sensors 56
Εικόνα 35 Grafana Dashboard 57
Εικόνα 36 Grafana Επιλογή Μεταβλητών 57
Εικόνα 37 Grafana DS18B20 και BH1750 58

Εικόνα 38 Grafana Create New Panel 58
Εικόνα 39 Grafana Panel Settings 59
Εικόνα 40 InfluxDB 60
Εικόνα 41 Grafana CSV 60
Εικόνα 42 Grafana Download CSV 61
Εικόνα 43 Θερμοκρασίες Αισθητήρων DHT11 63
Εικόνα 44 Υγρασίες Αισθητήρων DHT11 64
Εικόνα 45 Θερμοκρασίες Αισθητήρων DS18B20 65
Εικόνα 46 Θερμοκρασίες DHT11 CSV 66
Εικόνα 47 Huge APP 67
Εικόνα 48 Διατηρούμενο Μήνυμα 67
Εικόνα 49 Εντολές Χρήστη Με Node-Red 68
Εικόνα 50 Επανασύνδεση WiFi 69
Εικόνα 51 BLE Επανασύνδεση 70
Εικόνα 52 Raspberry Pi Imager 118
Εικόνα 53 Αρχείο ssh 119
Εικόνα 54 Putty 121
Εικόνα 55 Ενεργοποίηση VNC στο Raspberry Pi 123
Εικόνα 56 Αδυναμία σύνδεσης VNC με το Raspberry Pi 124
Εικόνα 57 Αλλαγή Ρυθμίσεων Ανάλυσης Οθόνης 125
Εικόνα 58 MQTT Explorer 126
Εικόνα 59 Dashboard-Storage Palette 129
Εικόνα 60 Ορισμός διεύθυνσης βάσης InfluxDB 134
Εικόνα 61 Ρυθμίσεις Συνδεσιμολογίας Με InfluxDB 135

Συντομογραφίες & Ακρωνύμια

IoT	Internet Of Things
BLE	Bluetooth Low Energy
MQTT	Queue Telemetry Transport
CPU	Central Processing Unit
MCU	Microcontroller Unit
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver-Transmitter
I2C	Inter-Integrated Circuit
I2S	Inter-IC Sound
SSH	Secure Socket Shell
VNC	Virtual Network Computing

1 Εισαγωγή

1.1 Πρόλογος

Η τεχνολογία Internet of Things αυξάνεται με ραγδαίους ρυθμούς τα τελευταία χρόνια και λαμβάνει όλο και περισσότερες θέσεις σε τομείς της καθημερινότητά μας. Όπως είναι φυσικό ένας από τους τομείς που επηρεάστηκαν, είναι αυτός της θέρμανσης και των οικιακών αυτοματισμών. Όσο όμως αυξάνονται οι «έξυπνες» συσκευές δημιουργείται μια πρόσθετη ανάγκη, αυτή της εκμετάλλευσης των μεγάλων ποσοτήτων δεδομένων που συλλέγουν. Δεν αρκεί πλέον η δημιουργία αυτόνομων προϊόντων θέρμανσης, αλλά έχει τεθεί η ανάγκη εκμετάλλευσης των δεδομένων προκειμένου να γίνει όλο το ενεργειακό σύστημα πιο έξυπνο, αποδοτικό, πιο φιλικό προς το περιβάλλον και πιο αξιόπιστο. Η χρησιμοποίηση δεδομένων από τους αισθητήρες σε όλα τα μέρη του δικτύου βοηθάει στην ανάλυση και πρόβλεψη της ζήτησης για την παραγωγή θέρμανσης, δημιουργώντας ένα πραγματικά ενεργειακά αποδοτικό σύστημα. [1]

1.2 Σκοπός της εργασίας

Σκοπός της διπλωματικής εργασίας είναι η δημιουργία ενός έξυπνου σύστηματος ελέγχου και απόδοσης της κεντρικής θέρμανσης κτηρίου. Η συγκεκριμένη υλοποίηση χρησιμοποιώντας μικροελεγκτές, απεξαρτοποιεί την λειτουργία του θερμοστάτη από την ανάγκη καλωδίωσης για την τροφοδοσία του και προσέφερει την δυνατότητα μετατροπής υπαρχόντων κτηρίων σε έξυπνα. Επιδεικνύεται επίσης η διαδικασία αποθήκευσης, ανάλυσης και οπτικοποίησης δεδομένων από δωρεάν εργαλεία τα οποία μπορούν να χρησιμοποιηθούν από τον οποιοδήποτε ανεξάρτητα από το επίπεδο προγραμματισμού που διαθέτει.

1.3 Δομή της εργασίας

Η εργασία αποτελείται από εφτά κεφάλαια τα οποία παρουσιάζονται ακολούθως:

Στο 1ο κεφάλαιο γίνεται μια σύντομη εισαγωγή και αναφέρεται ο σκοπός της διπλωματικής εργασίας. Επιπλέον αναλύονται τα στάδια και η δομή της.

Στο 2ο κεφάλαιο παρουσιάζονται οι προδιαγραφές και ο σχεδιασμός του συστήματος ελέγχου απόδοσης κεντρικής θέρμανσης. Αναφέρονται οι απαιτήσεις λειτουργίας και διερευνάται ο τρόπος λειτουργίας του.

Στο 3ο κεφάλαιο παρουσιάζονται οι θεωρητικές έννοιες των δικτύων και των πρωτοκόλλων επικοινωνίας που χρησιμοποιήθηκαν στην υλοποίηση του συστήματος. Δίνεται ιδιαίτερη έμφαση στην δομή και στον τρόπο μεταφοράς των μηνυμάτων.

Στο 4ο κεφάλαιο παρουσιάζεται το σύστημα αισθητήρων που υλοποιήθηκε. Αναλύονται τα υλικά μέρη που το αποτελούν και ο τρόπος διασύνδεσης μεταξύ τους.

Στο 5ο κεφάλαιο παρουσιάζεται ο διακομιστής και αναλύονται τα υλικά και λογισμικά μέρη που τον αποτελούν.

Στο 6ο κεφάλαιο παρουσιάζονται οι εφαρμογές διεπαφής χρήστη, δίνοντας έμφαση στον τρόπο που οπτικοποιούνται και αναλύονται τα δεδομένα.

Στο 7ο κεφάλαιο αναπτύσσονται τα αποτελέσματα και τα συμπεράσματα που προέκυψαν από τη χρήση του συστήματος και διατυπώνονται ιδέες για μελλοντικές επεκτάσεις ή βελτιώσεις προκειμένου να εξελιχθεί το σύστημα.

2 Έξυπνο Σύστημα Ελέγχου Απόδοσης Κεντρικής Θέρμανσης Κτηρίων

2.1 Το Internet of Things σαν σύστημα αυτοματισμού

Το 1999 ο Βρετανός Kevin Ashton, πρωτοπόρος σε θέματα τεχνολογίας επινόησε τον όρο «Internet of Things – IoT» ώστε να περιγράψει ένα προηγμένο σύστημα αυτοματισμού στο οποίο το Internet συνδέεται με τον φυσικό κόσμο εκμεταλλευόμενο τη δικτύωση, την ανίχνευση, τα μεγάλα δεδομένα και την τεχνολογία τεχνητής νοημοσύνης.

Αυτά τα συστήματα μπορούν να εφαρμοστούν σε όλους τους κλάδους μέσω της μοναδικής ευελιξίας τους, προσδίδοντας μεγαλύτερη διαφάνεια, έλεγχο και απόδοση. Ενισχύουν τη συλλογή δεδομένων, επιτρέποντας έτσι στους χρήστες να επιτύχουν αυτοματισμό, βαθύτερη ανάλυση και ενσωμάτωση και τελικά την βέλτιστη χρήση ενός συστήματος.

Το IoT εκμεταλλεύεται τις πρόσφατες εξελίξεις στο λογισμικό, τη μείωση των τιμών του υλικού και τη σύγχρονη στάση απέναντι στην τεχνολογία. Τα νέα και προηγμένα στοιχεία του φέρνουν σημαντικές αλλαγές στην παράδοση προϊόντων, αγαθών και υπηρεσιών. και τον κοινωνικό, οικονομικό και πολιτικό αντίκτυπο αυτών των αλλαγών. [2]

Όπως κάθε τεχνολογία, έτσι και το IoT, έχει κάποια πολύ σημαντικά πλεονεκτήματα, αλλά κρίβει και ορισμένους κινδύνους που εμφανίζονται σε πολλούς τομείς της καθημερινής μας ζωής.

Πλεονεκτήματα

- Βελτιστοποίηση: Η εφαρμογή και η χρήση του IoT συμβάλλει στην βελτίωση της χρήσης των συσκευών, της εμπειρίας των χρηστών, και τελικά στην βελτίωση της ποιότητας ζωής των χρηστών. Το IoT ξεκλειδώνει έναν κόσμο κρίσιμων λειτουργικών δεδομένων και δεδομένων πεδίου.
- Αποδοτικότητα: Το IoT συλλέγει άμεσα και ταχύτατα τις απαιτούμενες πληροφορίες. Αυτό βοηθά στην αποτελεσματικότητα της διαχείρισης των πόρων που χρησιμοποιούνται και στην μείωση της σπατάλης αυτών. Προσφέρουν δηλαδή μεγαλύτερη ακρίβεια.
- Βελτιωμένη συλλογή δεδομένων: Με το IoT ο χρήστης μπορεί να συλλέξει μεγαλύτερη ποσότητα δεδομένων με μεγαλύτερη ακρίβεια και αυτοματοποιημένο

τρόπο. Αυτό προσφέρει καλύτερη εικόνα της κατάστασης ή του θέματος που μελετάται.

Μειονεκτήματα

- Ασφάλεια: Το IoT δημιουργεί ένα οικοσύστημα συνεχώς συνδεδεμένων συσκευών που επικοινωνούν μέσω δικτύου. Το σύστημα προσφέρει μικρό έλεγχο παρά τα μέτρα ασφαλείας. Αυτό αφήνει τους χρήστες εκτεθειμένους σε διάφορα είδη δικτυακών επιθέσεων.
- Απόρρητο: Μέρος τις λειτουργίας των έξυπνων συσκευών IoT είναι να παρέχουν και να αναλύουν μεγάλες ποσότητες δεδομένων. Σε αρκετές περιπτώσεις τα προσωπικά δεδομένα χρηστών μπορεί να αποτελλούν μέρος αυτών των δεδομένων και χωρίς να υπάρχει η ενεργή συμμετοχή του χρήστη.
- Πολυπλοκότητα: Παρόλο που η τεχνολογία στον συγκεκριμένο τομέα προχωράει με γοργούς ρυθμούς πολλές από τις συσκευές δημιουργούν πολυπλοκότητα όσο αφορά την συντήρησή τους. Συμβαίνουν συχνές αποσυνδέσεις και δυσκολίες στην ένωση με το τοπικό δίκτυο ειδικά σε περιπτώσεις που το δίκτυο κατακλύζεται με μεγάλο όγκο συσκευών.
- Ευελιξία: Πολλές από τις εταιρίες που χρησιμοποιούν την τεχνολογία IoT, δημιουργούν προϊόντα τα οποία δεν συνεργάζονται με αντίστοιχα προϊόντα άλλων εταιριών. Αυτό δημιουργεί πρόβλημα στους καταναλωτές.

2.2 Ο ρόλος του Internet Of Things στους οικιακούς αυτοματισμούς

Τα τελευταία χρόνια η αύξηση στην κατανάλωση ενέργειας βρίσκεται στο επίκεντρο των κυβερνήσεων παγκοσμίως, κυρίως λόγω των επιπτώσεων στο περιβάλλον και συγκεκριμένα στην κλιματική αλλαγή. Ο βασικός στόχος είναι να υπάρξει μια αλλαγή όχι μόνο στα καύσιμα που χρησιμοποιούνται (από ορυκτά καύσιμα σε ανανεώσιμες πηγές ενέργειας), αλλά και στον τρόπο που αντιλαμβανόμαστε και διαχειριζόμαστε την ενέργεια στις υποδομές, τις μεταφορές (τόσο ανθρώπων όσο και εμπορευμάτων), στις γραμμές παραγωγής και φυσικά σε όλες τις εκφάνσεις τις προσωπικής μας ζωής [3], [4].

Μια από τις βασικότερες ανησυχίες έγκειται στην διαρκώς αυξητική κατανάλωση ενέργειας των πόλεων. Για παράδειγμα, το 40% της συνολικής ενέργειας που απαιτεί η Ευρώπη,

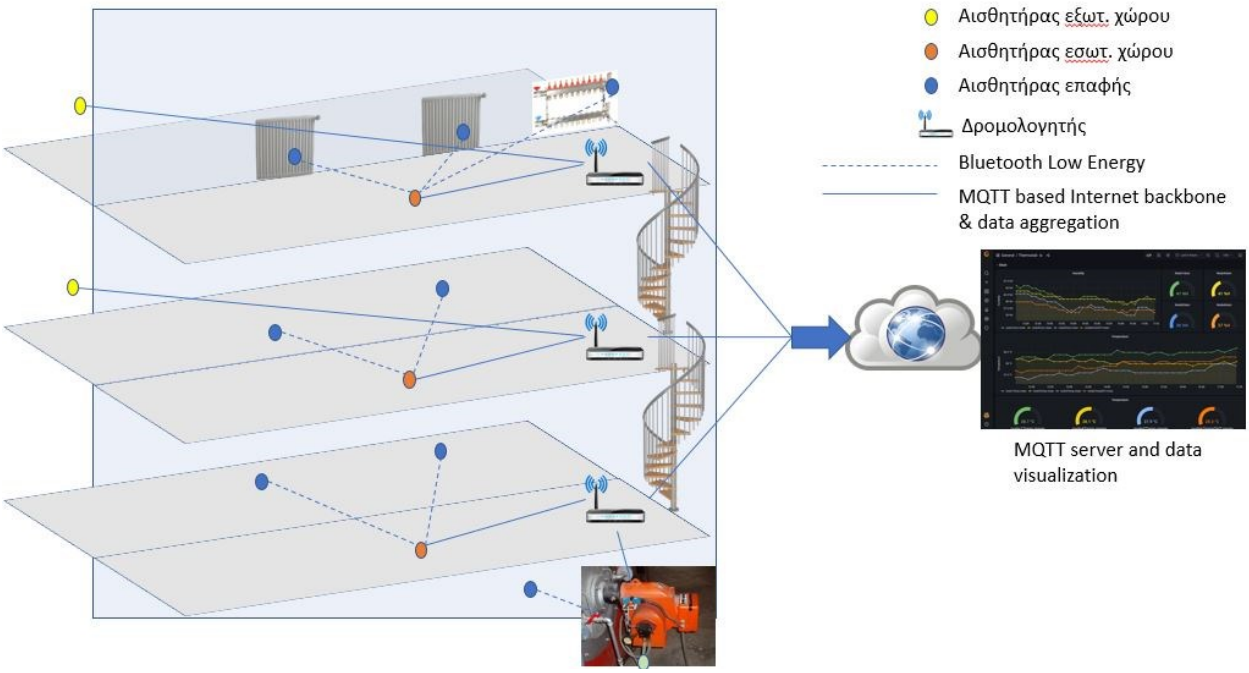
προέρχεται από τα κτήρια των πόλεων (τόσο γραφεία όσο και κατοικίες), και συγκεκριμένα απο τα συστήματα θέρμανσης, εξαερισμού και κλιματισμού (Heating, Ventilation and Air Conditioning systems ή εν συντομία HVAC). Τα συστήματα αυτά είναι ζωτικής σημασίας για όλους τους ανθρώπους αλλά η τεχνολογία στην οποία βασίζονται δεν είναι ακόμα τέτοια ώστε να τους επιτρέπει να μας προσφέρουν τις ανέσεις που χρειαζόμαστε, αλλά με τον πιο αποδοτικό, συμφέρον, και έξυπνο τρόπο [5], [6].

Αυτό είναι κάτι που μπορεί να αλλάξει με την τεχνολογία του Internet of Things. Τόσο οι κυβερνήσεις όσο και οι δήμοι, όπως και οι χρήστες μεμονωμένα, χρησιμοποιώντας το IoT, μπορούν να έχουν διπλό κέρδος αφού μειώνουν τα κόστη ενώ παράλληλα βελτιώνουν τις παρεχόμενες υπηρεσίες. Για την επίτευξη αυτής της αλλαγής απαιτούνται αλγόριθμοι που ελέγχουν την λειτουργία αυτών των συστημάτων και φυσικά την ανανέωση της ξεπερασμένης τεχνολογίας, με πιο σύγχρονη. Η σύγχρονη τεχνολογία επιτρέπει την ενσωμάτωση αισθητήρων και μικροελεγκτών στις κεντρικές μονάδες (θέρμανσης, εξαερισμού, κλιματισμού κτλ.), αλλά και σε διάφορα σημεία των κτηρίων, οι οποίες βελτιστοποιούν την χρήση των διάφορων συστημάτων με την χρήση των αλγορίθμων που προαναφέρθηκαν. Επιπλέον, όλες αυτές οι συσκευές (αισθητήρες, ελεγκτές) επικοινωνούν μεταξύ τους, αλλά και με την κεντρική μονάδα/πλακέτα ελέγχου ασύρματα μέσω δικτύων και Bluetooth. Με αυτό τον τρόπο, διασφαλίζεται η γρήγορη, και βέλτιστη λειτουργία και επικοινωνία του συστήματος, αλλά και η αποθήκευση δεδομένων.

Γίνεται λοιπόν αντιληπτό, ότι το έξυπνο κτήριο ή σπίτι είναι ακριβώς «έξυπνο» γιατί μπορεί να αυτό-ρυθμίζεται. Είναι δυνατόν να μειωθούν οι ενεργειακές ανάγκες καθώς για παράδειγμα μπορεί ένας όροφος ή ορισμένα δωμάτια ενός ορόφου να μην έχουν θέρμανση/κλιματισμό ή να έχουν χαμηλότερη/υψηλότερη θερμοκρασία ανάλογα με τις ανάγκες των ενοίκων. Αυτό φυσικά μπορεί να ξεφύγει από τα στενά όρια της θερμοκρασίας ενός σπιτιού (θέρμανση/κλιματισμός) και να επεκταθεί και σε άλλες δραστηριότητες που συμβάλλουν στην κατανάλωση ενέργειας. Για παράδειγμα, οι κουρτίνες μπορούν να ανοιγοκλείσουν και φωτισμός να αυξηθεί η να μειωθεί ανάλογα της φωτεινότητας στον χώρο. Αυτοί λοιπόν είναι οι λόγοι που το Internet of Things και οι έξυπνες συσκευές έχουν εισέλθει τόσο δυναμικά στις ζωές μας και αυτός είναι και ο κύριος σκοπός αυτής της εργασίας: να μελετηθεί πως γίνεται να μετατρέψουμε ένα συμβατικό σύστημα θέρμανσης, σε ένα σύγχρονο, έξυπνο σύστημα θέρμανσης.

2.3 Προδιαγραφές και σχεδιασμός συστήματος ελέγχου απόδοσης κεντρικής θέρμανσης

Το σύστημα που έχει υλοποιηθεί σε αυτή την εργασία έχει κύριο στόχο την καταμέτρηση, την ανάλυση και την δυνατότητα χειρισμού της κεντρικής θέρμανσης. Το σύστημα αυτό μπορεί να τοποθετηθεί σε ένα πολυώροφο κτήριο και η κύρια απαίτηση για την εφαρμογή του είναι να υπάρχει πρόσβαση σε δίκτυο WiFi σε όλα τα μέρη του κτηρίου που ανήκουν οι κεντρικοί αισθητήρες του συστήματος θέρμανσης.



Εικόνα 1 Σύστημα Κεντρικής Θέρμανσης Κτηρίου

Κάθε όροφος διαθέτει ένα πλήθος αισθητήρων, οι οποίοι είναι συνδεδεμένοι με μικροελεγκτές που παρέχουν δυνατότητες ασύρματης σύνδεσης. Το σύστημα δεν απαιτεί την προ-υπάρχουσα καλωδίωση για την τοποθέτησή του, δηλαδή έχει την δυνατότητα να τοποθετηθεί σε οποιοδήποτε κτήριο. Συνεπώς, όλες οι πλακέτες με μικροελεγκτές είναι αυτόνομες ενεργειακά διαθέτοντας μπαταρία.

Το σύστημα χωρίζεται σε 4 υποσυστήματα:

1. Υποσύστημα Εσωτερικών Χώρων: Το συγκεκριμένο υποσύστημα προϋποθέτει έναν αισθητήρα επαφής σε κάθε όροφο. Ο αισθητήρας, θα είναι τοποθετημένος στον κεντρικό σωλήνα ροής θερμού νερού προς τα θερμαντικά σώματα του ορόφου. Κάθε θερμαντικό σώμα με την σειρά του θα έχει τοποθετημένο από έναν αισθητήρα

επαφής αντίστοιχο με αυτόν του σωλήνα. Τέλος, στον κάθε όροφο θα υπάρχει ένας αισθητήρας για την μέτρηση της θερμοκρασίας και της υγρασίας του χώρου. Ωστόσο, ανάλογα με την διαμόρφωση του ορόφου, μπορούμε να τοποθετήσουμε έναν δεύτερο αισθητήρα υγρασίας και θερμοκρασίας προκειμένου να έχουμε μια πιο σφαιρική εικόνα για την θερμοκρασία/υγρασία του χώρου. Όπως είναι φυσικό, σε ένα τέτοιο υποσύστημα δεν θα μπορούσε να είναι δυνατή η διασύνδεση των αισθητήρων με την ίδια πλακέτα μικροελεγκτή, καθώς αυτό θα απαιτούσε πλειάδα καλωδίων. Για την αποφυγή χρήσης καλωδίων, ο κάθε αισθητήρας είναι τοποθετημένος σε διαφορετικό μικροελεγκτή και επικοινωνούν μεταξύ τους μέσω Bluetooth Low Energy. Ένας από τους συγκεκριμένους μικροελεγκτές του κάθε ορόφου έχει την δυνατότητα σύνδεσης στο WiFi για να μπορούν να στέλνονται τα δεδομένα σε κάποιο cloud ή σε κάποιο προσωπικό διακομιστή. Η χρησιμοποίηση πρωτοκόλλου BLE στους αισθητήρες εσωτερικού χώρου, προσφέρει χαμηλότερη κατανάλωση ενέργειας και το κάνει ιδανικό για συσκευές με μπαταρία.

2. Υποσύστημα Εξωτερικών Χώρων: Αυτό το υποσύστημα αποτελείτε από μια πλακέτα μικροελεγκτή στον κάθε όροφο. Η πλακέτα διαθέτει έναν αισθητήρα θερμοκρασία/υγρασίας και έναν αισθητήρα μέτρηση της ηλιακής φωτεινότητας. Η συγκεκριμένη πλακέτα θα συνδέεται απευθείας στο WiFi. Η υλοποίηση της, είναι ανεξάρτητη και μπορεί να τοποθετηθεί και στην ταράτσα του κτηρίου, αν βέβαια είναι δυνατή η σύνδεση στο WiFi.

3. Υποσύστημα Λεβητοστασίου: Το σύστημα αυτό απευθύνεται στον χώρο που ανήκει ο λέβητας του κτηρίου και είναι υπεύθυνος για την λειτουργία της θέρμανσης. Στον συγκεκριμένο χώρο τοποθετείται μια πλακέτα με μικροελεγκτή που διαθέτει έναν αισθητήρα επαφής για την μέτρηση της θερμοκρασίας του σωλήνα ζεστού νερού, καθώς ξεκινάει από τον λέβητα. Επίσης, έχει έναν αισθητήρα για την καταμέτρηση της θερμοκρασίας/υγρασίας του χώρου και ένα relay με το οποίο ο χρήστης ελέγχει τον λέβητα, δηλαδή το άνοιγμα και το κλείσιμο της θέρμανσης. Απαραίτητη και στην συγκεκριμένη περίπτωση είναι η ύπαρξη WiFi στον χώρο, ωστόσο ο συγκεκριμένος μικροελεγκτής είναι ο μοναδικός που θα τροφοδοτείται με καλωδίωση αφού θα συνδέεται με τον λέβητα.

4. Υποσύστημα αποθήκευσης δεδομένων: Το σύστημα αυτό απευθύνεται στον τρόπο αποθήκευσης και επεξεργασίας των δεδομένων. Υπάρχουν πολλές επιλογές

ανάλογα τις απαιτήσεις του χρήστη του συστήματος. Μπορεί να χρησιμοποιηθεί μια έτοιμη πλατφόρμα cloud, ένας απομακρυσμένος διακομιστής ή ένας τοπικός διακομιστής εντός του κτηρίου που υλοποιείται το σύστημα θέρμανσης.

Πρέπει να τονιστεί ότι η συγκεκριμένη διπλωματική εργασία δεν υλοποιεί ολόκληρο το σύστημα της κεντρικής θέρμανσης ενός κτηρίου, καθώς το κόστος για την αγορά του εξοπλισμού θα ήταν υψηλό. Ωστόσο υλοποιήθηκε το σύστημα ενός ορόφου και του λεβητοστασίου με δυνατότητες προέκτασης αρκεί να γίνουν οι απαραίτητες προσθήκες. Συγκεκριμένα υλοποιήθηκε ένα υποσύστημα εξωτερικού χώρου με ένα ελεγκτή και όσους αισθητήρες διαθέτει. Ένα υποσύστημα εσωτερικού χώρου για έναν όροφο με αισθητήρες για την καταμέτρηση ενός θερμαντικού σώματος, του σωλήνα τροφοδοσίας νερού και δύο αισθητήρες μέτρησης θερμοκρασίας/υγρασίας. Όλοι οι αισθητήρες συνδέονται με τους απαραίτητους μικροελεγκτές. Τέλος το υποσύστημα του λέβητα υλοποιήθηκε με τον προαναφερθέντα τρόπο. Επιπλέον λόγω της εποχής που πραγματοποιήθηκε η εργασία (καλοκαίρι) το σύστημα δεν τοποθετήθηκε σε λειτουργικό δίκτυο κεντρικής θέρμανσης αλλά χρησιμοποιήθηκε για την συλλογή, ανάλυση και επεξεργασία των δεδομένων που συλλέγονται από τους αισθητήρες. Για περισσότερες λεπτομέρειες του συστήματος υλοποίησης αναφέρονται στο κεφάλαιο 4.

2.3.1 Απαιτήσεις Συστήματος

Το σύστημα ελέγχου της κεντρικής θέρμανσης πρέπει να εκτελεί τις παρακάτω λειτουργίες:

- Θεωρείται δεδομένη η ύπαρξη δικτύου WiFi και όλες οι μετρήσεις των αισθητήρων να είναι διαθέσιμες σε αυτό.
- Ο ρυθμός δειγματοληψίας να παίρνει τιμές από 1 λεπτό ως 15 λεπτά και να είναι μία παράμετρος που δίνεται από το χρήστη.
- Όλοι οι αισθητήρες εκτός του λεβητοστασίου πρέπει να διαθέτουν μπαταρία και η επέμβαση στον χώρο να είναι ελάχιστη.
- Η μετάδοση των δεδομένων να γίνεται ασύρματα, χωρίς την χρήση καλωδίων.
- Οι μετρήσεις δεν χρειάζεται να έχουν μεγάλη ακρίβεια.
- Θα πρέπει να υπάρχει δυνατότητα αποθήκευσης των δεδομένων και ανάκτησή τους όταν χρειάζεται.

- Να υπάρχει δυνατότητα δημιουργίας CSV αρχείου με όλα τα επιθυμητά δεδομένα από τον διακομιστή για περαιτέρω επεξεργασία τους.
- Δυνατότητα χειρισμού της θέρμανσης διαμέσου κάποιου περιβάλλοντος διεπαφής χρήστη.
- Δυνατότητα γραφικής απεικόνισης και ανάλυσης των μετρήσεων σε περιβάλλον διεπαφής χρήστη.
- Τα λογισμικά που χρησιμοποιούνται να είναι δωρεάν, εύκολα στην εκμάθηση, εύκολα στον χειρισμό και χωρίς την προυπόθεση γνώσεων προγραμματισμού.
- Να μπορεί να υλοποιηθεί και τοπικά με την χρήση μόνο τοπικού δικτύου.

2.3.2 Σχεδιασμός συστήματος

Ένα από τα δυσκολότερα σημεία της συγκεκριμένης εργασίας είναι η επιλογή των υλικών και λογισμικών που απαιτούνται για την υλοποίηση του συστήματος ελέγχου της κεντρικής θέρμανσης. Τόσο η επιλογή των ελεγκτών, των αισθητήρων, των περιφερειακών υλικών τροφοδοσίας και διασύνδεσης, καθώς του διακομιστή και των λογισμικών που τον αποτελούν προήλθε μετά από μεγάλη έρευνα. Τα υλικά που χρησιμοποιήθηκαν αγοράστηκαν είτε από την Κίνα είτε από την Ελλάδα. Την συγκεκριμένη περίοδο υπήρχε μια μεγάλη αύξηση τιμών σε πολλά από τα είδη αισθητήρων και κυρίως σε προϊόντα που διαθέτουν chip. Επίσης, εξαιτίας των συνθηκών που επικρατούσαν στις μεταφορές (κατά την διάρκεια της πανδημίας που προκάλεσε η νόσος COVID), η προμήθεια προϊόντων από την Κίνα διαρκούσε μήνες και αυτός ήταν κατασταλτικός παράγοντας για να ψωνίσεις από εκεί. Τέλος, στην ελληνική αγορά υπήρχε έλλειψη σε πολλά από τα προϊόντα που είχαν οριστεί στον αρχικό σχεδιασμό του συστήματος.

Όλοι οι παραπάνω λόγοι οδήγησαν στην αναγκαστική προσαρμογή του συστήματος με βάση τα υλικά που τελικά αποκτήθηκαν έχοντας ως κύριο παράγοντα το κόστος της υλοποίησης.

Τα υλικά υλοποίησης συστήματος:

Για την μέτρηση της θερμοκρασίας και υγρασίας του χώρου χρησιμοποιήθηκε ο αισθητήρας DHT11. Αν και δεν είναι ακριβείας και δεν είναι ιδανικός για την συγκεκριμένη εργασίας μιας και δεν μετρά αρνητικές τιμές θερμοκρασίας, ωστόσο ήταν ο οικονομικότερος της αγοράς. Επιπλέον πρόσθετος λόγος επιλογής είναι ότι ο κώδικας με τον οποίο υλοποιήθηκε μπορεί να χρησιμοποιηθεί και για τον DHT22, που εξαιτίας του κόστους του δεν αγοράστηκε, ωστόσο όποιος επιθυμεί μπορεί να αντικαταστήσει τον DHT11 με τον DHT22 για να πετύχει μεγαλύτερη ακρίβεια και να επιβλέπει αρνητικές τιμές θερμοκρασίας.

Για την μέτρηση της θερμοκρασίας με επαφή χρησιμοποιήθηκε ο αισθητήρας DS18B20 waterproof. Ο συγκεκριμένος αισθητήρας είναι αρκετά οικονομικός και ιδανικός για την λειτουργία που απαιτείται από την εργασία.

Για την μέτρηση της ηλιακής φωτεινότητας χρησιμοποιήθηκε ο αισθητήρας BH1750. Με χαμηλό κόστος και μεγάλη ακρίβεια ήταν ο ιδανικός για το σύστημά. Επιπλέον, η χρήση του πρωτοκόλλου I2C, κάνει πολύ εύκολη την υλοποίησή του.

Για το άνοιγμα και τον κλείσιμο της θέρμανσης χρησιμοποιήθηκε relay τεσσάρων καναλιών.

Ως μικροελεγκτή (MCU) αποφασίστηκε να χρησιμοποιηθούν πλακέτες που διαθέτουν τον ESP32. Ο συγκεκριμένος μικροελεγκτής διαθέτει κεραίες WiFi και Bluetooth με αποτέλεσμα να τον κάνουν ιδανικό για την εργασία. Από τις διάφορες υλοποιήσεις που κυκλοφορούν στην αγορά επιλέχτηκαν τέσσερις wemos lolin32 οι οποίες διαθέτουν δυνατότητα σύνδεσης της συσκευής με μπαταρίας χωρίς να απαιτούνται πρόσθετες υλοποιήσεις. Οι συγκεκριμένες πλακέτες είναι ιδανικές για το υποσύστημα εσωτερικού χώρου. Επιπλέον χρησιμοποιήθηκαν δύο πλακέτες Doit ESP32 devkit v1 τόσο για το υποσύστημα εξωτερικού χώρου όσο και για το υποσύστημα του λεβητοστασίου.

Ως διακομιστής χρησιμοποιήθηκε ένα raspberry pi 4 με 4GB ram. Οι λόγοι επιλογής μιας δικιάς μας υλοποίησης διακομιστή και κυρίως raspberry pi είναι οι εξής:

- Τα αντίστοιχα server cloud (π.χ. thingspeak) στην δωρεάν μορφή τους προσφέρουν περιορισμένες δυνατότητες οπτικοποίησης ή έχουν περιορισμένο χρονικό περιθώριο χρησιμοποίησής τους. Κάτι τέτοιο δεν ήταν αποδεκτό επειδή

χρησιμοποιήθηκαν πολλές πλακέτες και απαιτούνται δυνατότητες επέκτασης της εφαρμογής.

- Όσο αφορά το υλικό κομμάτι επιλέχτηκε το raspberry pi 4 επειδή προσφέρει μια οικονομική λύση για την υλοποίηση ενός διακομιστή. Επιπλέον η κατανάλωση ρεύματος δεν ξεπερνά τα 5 Watt, οπότε δεν αποτελεί βάρος η πολύωρη λειτουργία του. Αν και η αρχική σκέψη ήταν η αγορά ενός raspberry pi 3 ωστόσο δεν υπήρχε απόθεμα στην ελληνική αγορά την συγκεκριμένη περίοδο.

Επιπλέον πρέπει να τονιστεί ότι στο raspberry pi 4 δεν έχει δώθει καμιά άλλη λειτουργικότητα στο σύστημα εκτός από αυτή του διακομιστή. Αυτή η επιλογή έγινε για να αποδειχθεί ότι ο διακομιστής είναι κάτι ανεξάρτητο από το υπόλοιπο σύστημα, και συνεπώς ο χρήστης μπορεί να χρησιμοποιήσει ως διακομιστή όποια συσκευή θέλει.

Παραπάνω λεπτομέρειες για τα υλικά αισθητήρων και για τον διακομιστή βρίσκονται στο κεφάλαιο 4 και στο κεφάλαιο 5.

3 Δίκτυα Και Πρωτόκολλα Επικοινωνίας

Στο συγκεκριμένο κεφάλαιο θα αναφερθούμε στο είδος της σύνδεσης και στα πρωτόκολλα επικοινωνίας που θα χρησιμοποιηθούν από το σύστημα προκειμένου να είναι δυνατή η επικοινωνία μεταξύ των μονάδων που το αποτελούν.

3.1 WIFI

Το WiFi είναι ένα πρότυπο ασύρματης επικοινωνίας που χρησιμοποιείται σε ασύρματα τοπικά δίκτυα (WLAN), και επιτρέπει σε ηλεκτρονικές συσκευές να ανταλλάσσουν δεδομένα ή να συνδέονται στο διαδίκτυο με τη χρήση ραδιοκυμάτων.

Η ιστορία του είναι μακρά και ενδιαφέρουσα. Το 1971, το Πανεπιστήμιο της Χαβάης συνέδεσε τα νησιά με μια πρωτοποριακή, για τότε, ασύρματη τεχνολογία. Το 1985 η Ομοσπονδιακή Επιτροπή Επικοινωνιών των Ηνωμένων Πολιτειών της Αμερικής (U.S Federal Communications Commission) απελευθέρωσε την χρήση των συχνοτήτων 2.4GHz, οι οποίες χρησιμοποιούνται από συσκευές όπως οι φούρνοι μικροκυμάτων. Το 1991 οι εταιρίες η NCR Corporation και AT&T Corporation εφηύραν το πρώτο ασύρματο δίκτυο με την ονομασία WaveLan, με σκοπό να χρησιμοποιηθεί σε ταμειακές μηχανές. Το 1992 και το 1996, ο Οργανισμός Επιστημονικής και Βιομηχανικής Έρευνας της Βρετανικής Κοινοπολιτείας (Commonwealth Scientific and Industrial Research Organisation) κατασκεύασε πατέντες για την χρήση του WiFi σε προϊόντα. Η πρώτη έκδοση του προτύπου δικτύωσης 802.11, εκδόθηκε το 1997 και το 1999 αναβαθμίστηκε σε 802.11b. [7]

Ο τρόπος λειτουργίας του WiFi βασίζεται στην χρήση ραδιοκυμάτων για τη μετάδοση δεδομένων σε ένα δίκτυο. Βασικό προαπαιτούμενο για την σύνδεση μίας συσκευής σε ένα δίκτυο Wi-Fi, είναι ένας ασύρματος προσαρμογέας (wireless adapter) ο οποίος μεταφράζει τα δεδομένα σε ραδιοκύματα. Στη συνέχεια αυτά τα ραδιοκύματα αποστέλλονται μέσω μίας κεραίας σε ένα αποκωδικοποιητή (router). Μετά την αποκωδικοποίηση, τα ραδιοκύματα αποστέλλονται στο διαδίκτυο μέσω της ενσύρματης σύνδεσης Ethernet. Επειδή ένα δίκτυο Wi-Fi υποστηρίζει την αμφίδρομη λειτουργία αποστολής δεδομένων, δεδομένα που προέρχονται από το διαδίκτυο θα περάσουν μέσω του αποκωδικοποιητή (router), όπου θα κωδικοποιηθούν σε ραδιοκύματα και θα ληφθούν από τον ασύρματο προσαρμογέα της συσκευής που είναι ενωμένη στο δίκτυο WiFi. Πλέον, τα ραδιοκύματα ενός δικτύου WiFi

μεταδίδονται στις συχνότητες 2,4 GHz ή 5 GHz. Οι συχνότητες αυτές είναι μεγαλύτερες απο τις συχνότητες που χρησιμοποιούνται για τα κινητά τηλέφωνα, ασυρμάτους, ραδιόφωνα και τηλεοράσεις. Το γεγονός αυτό επιτρέπει στα ραδιοκύματα να μεταφέρουν περισσότερα δεδομένα.

3.2 Bluetooth Low Energy (BLE)

Το Bluetooth Low Energy (BLE για συντομία) είναι μια τεχνολογία, η οποία αναπτύχθηκε το 2010 από το Bluetooth Special Interest Group (SIG) σαν συμπληρωματική τεχνολογία του κλασικού Bluetooth 4.0 [8]

Η κλασσική έκδοση Bluetooth υποστηρίζει ένα σύστημα που ονομάζεται Basic Rate(BR), συχνά αναφερόμενο ώς Basic Rate/Enhanced Data Rate (BR/EDR). Ο κύριος λόγος σχεδιασμού του BLE είναι να βρεθεί μια παραλλαγή της έκδοσης Bluetooth με την ελάχιστη κατανάλωση, προσφέροντας χαμηλού εύρους βελτιστοποίηση, βοηθώντας έτσι την δημιουργία εφαρμογών χαμηλού κόστους. Έτσι λοιπόν, το BLE σχεδιάστηκε για να μεταδίδει πολύ μικρά πακέτα δεδομένων σε χαμηλό εύρος ζώνης (low bandwidth), καταναλώνοντας συγχρόνως λιγότερη ενέργεια από παρόμοιες BR/EDR συσκευές. Συγκεκριμένα, είναι σχεδιασμένο έτσι ώστε να υποστηρίζει αποτελεσματικά εφαρμογές με περιορισμένη ενέργεια και πυρίτιο, διευκολύνοντας τες ώστε να λειτουργούν για μια μεγάλη περίοδο χρόνου χρησιμοποιώντας μπαταρία [9].

Η βασική διαφορά με το «συμβατικό» Bluetooth, είναι ότι το BLE υποστηρίζει όχι μόνο επικοινωνία από σημείο σε σημείο (point-to-point), αλλά λειτουργία μετάδοσης (broadcast mode) και δίκτυο πλέγματος (mesh network).

- **Point to point**: Σε αυτή την λειτουργία δημιουργείται άμεση σύνδεση μεταξύ του διακομιστή (server) και του πελάτη (client) για την ανταλαγή δεδομένων.
- **Broadcast mode**: Σε αυτή τη λειτουργία, ένας διακομιστής μεταδίδει δεδομένα σε πολλούς δέκτες/πελάτες που είναι συνδεδεμένοι με τον server.
- **Mesh network**: Σε αυτή τη λειτουργία όλες οι συσκευές είναι μεταξύ τους συνδεδεμένες (λεγόμενη και σύνδεση many to many).

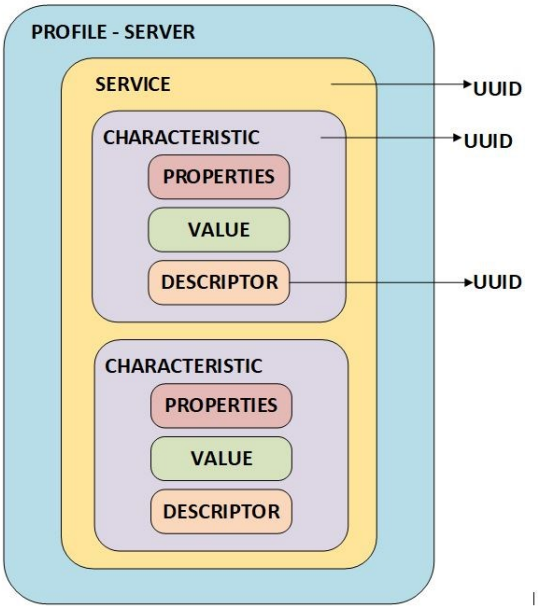
Με το Bluetooth Low Energy υπάρχουν δύο τύποι συσκευών, ο BLE Server και ο BLE Client. Οι συσκευές που υλοποιούν την λειτουργικότητα του BLE Server διαφημίζουν την

ύπαρξή τους ώστε να μπορούν να βρεθούν από τους BLE Client και να μεταδώσουν τα δεδομένα που έχουν συλλέξει. Ο BLE Client σαρώνει τις κοντινές συσκευές και όταν βρίσκει τον διακομιστή που αναζητά, δημιουργεί μια σύνδεση για να λάβει τα δεδομένα που χρειάζεται. Από την στιγμή που θα δημιουργηθεί αυτή η σύνδεση, η διαδικασία της διαφήμισης από την μεριά του BLE Server θα σταματήσει και δεν θα υπάρχει δυνατότητα να συνδεθεί μαζί του άλλη συσκευή.

Γενικά στο πρωτόκολλο BLE ένας Server μπορεί να συνδεθεί μόνο με έναν BLE Client, αντίθετα ο BLE Client μπορεί να συνδεθεί με όσους BLE Server επιθυμεί ταυτόχρονα και να συλλέγει τα δεδομένα τους. Επιπλέον δύο BLE Server δεν έχουν την δυνατότητα να διαμοιράζονται μεταξύ τους δεδομένα, ούτε την δυνατότητα να ξεκινήσουν μια σύνδεση. Τις συνδέσεις τις δημιουργούν πάντα οι BLE Client με τους αντίστοιχους Server που εντοπίζουν διαμέσου της διαδικασίας της διαφήμισης. Αν ένας BLE Server δεν έχει ενεργοποιημένη την διαδικασία της διαφήμισης δεν θα μπορέσει ποτέ κάποιος BLE Client να συνδεθεί μαζί του.

Στο πρωτόκολλο BLE τα δεδομένα διαδίδονται μεταξύ των συσκευών χρησιμοποιώντας μια ιεραρχική δομή με την ονομασία GATT (Generic Attribute Profile). Η συγκεκριμένη δομή καθορίζει τον τρόπο με τον οποίο οι συσκευές BLE στέλνουν και λαμβάνουν τα μηνύματα. Η κατανόηση αυτής της ιεραρχίας είναι σημαντική γιατί διευκολύνει στην κατανόηση του τρόπου χρήσης του πρωτοκόλλου BLE και την σύνταξη των εφαρμογών που μπορούν να εφαρμοστούν με αυτό.

Η παρακάτω εικόνα, αναπαριστά την ιεραρχική δομή των μηνυμάτων στο πρωτόκολλο BLE:



Εικόνα 2 Ιεραρχική Δομή BLE Μηνυμάτων

Στο ανώτερο επίπεδο της ιεραρχίας έχουμε το «Profile». Με τον όρο «Profile» αναφερόμαστε στον ίδιο τον BLE Server, από το συγκεκριμένο γεγονός καταλαβαίνουμε ότι την ιεραρχική δομή και τις ιδιότητές της, την καθορίζει ο BLE Server. Κάθε BLE Server περιέχει μια ή περισσότερες υπηρεσίες (Service). Κάθε «Service» με την σειρά του περιέχει ένα ή περισσότερα χαρακτηριστικά (Characteristic). Τα συγκεκριμένα «Characteristic» μπορεί να έχουν κανένα ή περισσότερα «Descriptor» [10].

Πιο συγκεκριμένα οι υπηρεσίες είναι μια συλλογή πληροφοριών όπως για παράδειγμα οι αναγνώσεις αισθητήρων. Υπάρχουν προκαθορισμένες υπηρεσίες για διάφορους τύπους δεδομένων που ορίζονται από το SIG (Bluetooth Special Interest Group) όπως το επίπεδο μπαταρίας, πίεση αίματος, καρδιακός ρυθμός κ.λπ.

Τα χαρακτηριστικά ανήκουν πάντα σε μια υπηρεσία (Service) και είναι το σημείο της ιεραρχικής δομής στο οποίο περιέχονται τα πραγματικά δεδομένα που θέλουμε να μεταδώσουμε. Το χαρακτηριστικό έχει πάντα δύο γνωρίσματα, την τιμή (value) και τις ιδιότητες (Properties) αυτής.

Με τον όρο ιδιότητες αναφερόμαστε στην περιγραφή του τρόπου αλληλεπίδρασης της τιμής του χαρακτηριστικού με τον BLE Client. Δηλαδή σε αυτό καθορίζονται οι λειτουργίες και οι διαδικασίες που μπορούν να χρησιμοποιηθούν με το χαρακτηριστικό, όπως Read, Write, Notify, Indicate και Broadcast.

Τέλος, κάτι που αξίζει να σημειωθεί είναι ότι τα «Service» , τα «Characteristic» και τα «Descriptor» έχουν ένα μοναδικό αριθμό που ονομάζεται UUID (Universally Unique Identifier). Το UUID αποτελείται από 128 bit (16 bytes) και χρησιμεύει στο να μπορούν να διακρίνονται τόσο τα «Service» όσο και τα «Characteristic» από τα υπόλοιπα. Έτσι σε ένα περιβάλλον με μια πληθώρα διαφημίσεων με την χρήση του UUID μπορούμε να εντοπίσουμε το χαρακτηριστικό μιας συγκεκριμένης υπηρεσίας που μας ενδιαφέρει, προκειμένου να επεξεργαστούμε την τιμή του [11].

Τα παραπάνω, έχουν καταστήσει την χρήση του BLE ως την πιο διαδεδομένη λύση για την ανάπτυξη ασύρματων εφαρμογών πολύ χαμηλής κατανάλωσης και ως εκ τούτου στο επίκεντρο του διαδικτύου των πραγμάτων (Internet of Things, ή για συντομία IoT). Αυτό φυσικά, δεν αποτελεί έκπληξη αφού με την πάροδο του χρόνου, η ζήτηση για «έξυπνες» συσκευές που έχουν πρόσβαση σε μικρά οικιακά και βιομηχανικά δίκτυα, τα οποία με τη σειρά τους αποκτούν πρόσβαση στο Ίντερνετ, ολοένα και μεγαλώνει. Για το λόγο αυτό, η ανάγκη για τη σύνδεση μικρών συσκευών με ελάχιστη παροχή ενέργειας (battery powered), οι οποίες έχουν πληθώρα αισθητήρων όλο και αυξάνεται.

3.3 Πρωτόκολλο Επικοινωνίας MQTT

Το πρωτόκολλο επικοινωνίας MQTT ή Queue Telemetry Transport όπως είναι το πλήρες όνομά του, είναι ένα ελαφρύ συμπαγές πρωτόκολλο που λειτουργεί πάνω από το από το πρωτόκολλο TCP/IP και βασίζεται σε μηνύματα δημοσίευσης/εγγραφής. Σχεδιάστηκε για την επικοινωνία μεταξύ συστημάτων (Machine to Machine ή εν συντομία M2M) από τους Andy Stanford-Clark της IBM και Arlen Nipper της Arcom το 1999.

Οι αρχικές εκδόσεις του πρωτοκόλλου χρησιμοποιήθηκαν εσωτερικά από την IBM. Το 2010 η έκδοση 3.1 δόθηκε ελεύθερα στο κοινό με άδεια royalty-free ενώ η έκδοση 3.1.1 έγινε standard από τον οργανισμό OASIS τον Οκτώβριο του 2014 [12].

Παρότι το σύστημα αναπτύχθηκε αρχικά για την αξιόπιστη σύνδεση και τηλεμετρία αγωγών πετρελαίου με δορυφόρο, παραμένει ακόμα εξαιρετικά ελκυστικό λόγω των χαρακτηριστικών πλεονοκτημάτων που προσφέρει. Συγκεκριμένα το πρωτόκολλο χρησιμοποιείται ευρέως σε πολλούς τομείς και εφαρμογές όπως για παράδειγμα στο διαδίκτυο των πραγμάτων (Internet of Things).

Το MQTT εξελίσσεται γρήγορα ως ένα από τα κύρια πρωτόκολλα ανάπτυξης εφαρμογών του Διαδικτύου των πραγμάτων εξαιτίας των πλεονεκτημάτων που διαθέτει [13]:

- Απλή υλοποίηση και επικοινωνία,
- Ικανότητα να λειτουργήσει χωρίς μεγάλη υπολογιστική ισχύ,
- Χρήση μικρού δικτυακού εύρους ζώνης,
- Επεκτασιμότητα, δηλαδή μπορεί να υποστηρίξει πολλαπλές συσκευές εύκολα και γρήγορα,
- Υποστήριξη διαφόρων επιπέδων ποιότητας υπηρεσιών (Quality of Service)

Οι περισσότερες βιβλιοθήκες MQTT ορίζουν ορισμένες τυπικές μεθόδους εντολών προκειμένου να γίνει δυνατή η επικοινωνία Client και Broker:

- Connect - δημιουργία σύνδεσης με το MQTT Broker
- Disconnect - διακοπή σύνδεσης με το MQTT Broker
- Publish - δημοσίευση δεδομένων σχετικά με κάποιο θέμα στον MQTT Broker
- Subscribe - Εγγραφή σε ένα θέμα στον MQTT Broker
- Unsubscribe - διαγραφή από το θέμα

Λειτουργία MQTT:

Το συγκεκριμένο πρωτόκολλο επικοινωνίας βασίζει την λειτουργία του στην ανταλλαγή των μηνυμάτων (messages) που πραγματοποιείται μεταξύ των πελατών (clients) και του διακομιστή (broker).

Όπως ίσως προκύπτει από το όνομα, ο διακομιστής λαμβάνει όλα τα μηνύματα από τους πελάτες και τα δρομολογεί ανάλογα προς τους πελάτες προορισμού. Πελάτης μπορεί να θεωρηθεί οποιαδήποτε συσκευή, που εκτελεί μια βιβλιοθήκη MQTT και συνδέεται μέσω ενός broker στο δίκτυο. Ο εκάστοτε πελάτης, μπορεί να κάνει δύο διαδικασίες: δημοσίευση - έκδοση (publish) ή/και εγγραφή - συνδρομή (subscribe) σε ένα θέμα (topic).

Σε αυτό το σημείο, πρέπει να διευκρινιστεί ότι κάθε μήνυμα έχει ένα θέμα (topic), και όλες οι πληροφορίες οργανώνονται με ιεραρχικό τρόπο (επίπεδα) χρησιμοποιώντας μια εμπρόσθια κάθετο ως διαχωριστή, παραδείγματος χάρη «home/node1/temp». Τα θέματα δημιουργούνται από τον συνδρομητή και τον εκδότη και δεν εκχωρούνται εκ των

προτέρων από τον MQTT Broker. Όσο ένα όνομα θέματος ακολουθεί τα πρότυπα ονομασίας, είναι αποδεκτό από τον Broker. Το όνομα του θέματος αποτελείται από μια συμβολοσειρά στον κώδικα UTF-8 και πρέπει να περιλαμβάνει τουλάχιστον ένα χαρακτήρα.

Συνεπώς όταν οι εκδότες διαθέτουν μηνύματα για διανομή, στέλνουν τα δεδομένα στον συνδεδεμένο broker. Αυτός με την σειρά του φιλτράρει τα μηνύματα με βάση το θέμα και στη συνέχεια τα διανέμει μόνο σε όσους πελάτες είναι συνδρομητές (ή εγγεγραμμένοι) στα συγκεκριμένα θέματα.

Αυτό που κάνει το πρωτόκολλο και την διάταξη χρήσιμη και εύκολη ως προς την υλοποίηση, είναι ότι ο εκδότης δεν χρειάζεται να έχει δεδομένα σχετικά με τον αριθμό ή τις επιμέρους τοποθεσίες των συνδρομητών, και οι συνδρομητές με τη σειρά τους δεν χρειάζεται να διαμορφώσουν τα δεδομένα σχετικά με τον κάθε εκδότη.

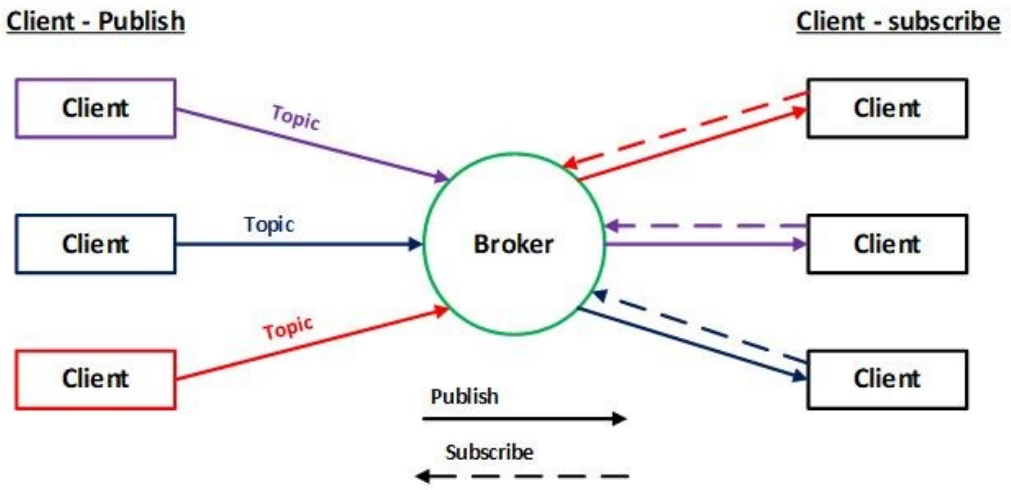
Οι πελάτες αλληλεπιδρούν μόνο με έναν διακομιστή, αλλά ένα σύστημα μπορεί να περιέχει αρκετούς διακομιστές, που ανταλλάσσουν δεδομένα με βάση τα θέματα των πελατών τους.

Σε γενικές γραμμές ο μεσίτης δεν αποθηκεύει μηνύματα, μόλις το μήνυμα αποστέλλεται στους συνδρομητές πελάτες, ο μεσίτης το διαγράφει. Σε περίπτωση που ένας διακομιστής λάβει ένα μήνυμα με θέμα για το οποίο δεν υπάρχουν τρέχοντες συνδρομητές, ο broker απορρίπτει το μήνυμα, εκτός και εάν ο εκδότης χαρακτηρίσει το μήνυμα ως «διατηρούμενο». Ένα τέτοιο μήνυμα δεν διαφέρει σε τίποτα από τα υπόλοιπα MQTT μηνύματα, εκτός από την σχετική σημαία που με την σειρά της πρέπει να είναι αληθής.

Αν το μήνυμα χαρακτηριστεί ως «διατηρούμενο μήνυμα» για το επιλεγμένο θέμα. Κάθε πελάτης που εγγράφεται σε αυτό, λαμβάνει το διατηρούμενο μήνυμα αμέσως μετά την εγγραφή του. Ο διακομιστής αποθηκεύει μόνο ένα «διατηρούμενο μήνυμα» ανά θέμα. Αυτό επιτρέπει στους νέους συνδρομητές του συγκεκριμένου θέματος, να μην εξαρτώνται απο το πότε θα στείλει την επόμενη ενημέρωση ο αντίστοιχος πελάτης, αλλά να λαμβάνουν την τελευταία καταχωρημένη τιμή. Αυτή την ιδιότητα την χρησιμοποιούμε στο σύστημα που θα υλοποιηθεί διαμέσου του Node-Red όπως θα διατυπωθεί στο κεφάλαιο 6.

Επιπλέον, όταν ένας πελάτης δημοσίευσης συνδέεται για πρώτη φορά με τον διακομιστή, μπορεί να ρυθμίσει ένα προεπιλεγμένο μήνυμα που θα αποστέλλεται στους συνδρομητές στην περίπτωση που ο διακομιστής εντοπίσει ότι ο πελάτης έχει αποσυνδεθεί απροσδόκητα από τον διακομιστή.

Η παρακάτω εικόνα, αναπαριστά την λειτουργία ενός MQTT Broker:



Εικόνα 3 Αναπαράσταση της διάταξης και της λειτουργίας του MQTT

Όπως φαίνεται, οι πελάτες που δημοσιεύουν (publish) τα δεδομένα στον διακομιστή (broker) MQTT ορίζουν ένα συγκεκριμένο θέμα στο μήνυμα τους. Στην συνέχεια, ο διακομιστής (broker) στέλνει τα δεδομένα στους πελάτες που αποτελούν συνδρομητές (subscribers) για το συγκεκριμένο θέμα. Αξίζει να σημειωθεί ότι οι πελάτες (subscribers) μπορούν να λαμβάνουν διάφορα δεδομένα από πολλούς publishers, ανάλογα με τη συνδρομή σε αντίστοιχα θέματα.

Τέλος, ένα από τα βασικά πλεονεκτήματα του πρωτοκόλλου MQTT είναι η υποστήριξη διαφόρων επιπέδων ποιότητας υπηρεσιών (Quality of Service ή εν συντομία QoS). Το επίπεδο ποιότητας των υπηρεσιών (QoS) είναι μια συμφωνία μεταξύ του αποστολέα ενός μηνύματος και του παραλήπτη του, που καθορίζει την εγγύηση παράδοσης για ένα συγκεκριμένο μήνυμα. Το MQTT επιτρέπει την επιλογή 3 επιπέδων ποιότητας υπηρεσιών και ο πελάτης έχει τη δυνατότητα να επιλέξει ένα επίπεδο υπηρεσιών που ταιριάζει με την αξιοπιστία του δικτύου και τη λογική εφαρμογής του.

Επειδή το MQTT διαχειρίζεται την εκ νέου μετάδοση μηνυμάτων και εγγυάται την παράδοση (ακόμη και όταν η υποκείμενη μεταφορά δεν είναι αξιόπιστη), το QoS διευκολύνει πολύ την επικοινωνία σε αναξιόπιστα δίκτυα. Τα διαθέσιμα επίπεδα είναι τα εξής [14]:

- **Επίπεδο QoS 0**: Το ελάχιστο επίπεδο QoS 0 δεν εγγυάται την παράδοση του μηνύματος. Ο παραλήπτης δεν χρειάζεται να επιβεβαιώσει με «request» για την λήψη του και έτσι ο αποστολέας δεν χρειάζεται να το στείλει ξανά σε περίπτωση μη παράδοσης. Είναι ιδανικό για εφαρμογές που μπορούν να ανεχτούν την απώλεια μυνημάτων, μιας και λόγο της διαδικασίας που ακολουθεί είναι το πιο γρήγορο.
- **Επίπεδο QoS 1**: Το επίπεδο ποιότητας 1 εγγυάται ότι ένα μήνυμα παραδίδεται τουλάχιστον μία φορά στον δέκτη. Ο αποστολέας αποθηκεύει το μήνυμα έως ότου λάβει μια επιβεβαίωνση από τον δέκτη για τη λήψη του μηνύματος. Είναι πιθανό ένα μήνυμα να σταλεί ή να παραδοθεί πολλές φορές.
- **Επίπεδο QoS 2**: Το QoS 2 είναι το υψηλότερο επίπεδο υπηρεσιών στο MQTT. Αυτό το επίπεδο εγγυάται ότι κάθε μήνυμα λαμβάνεται μόνο μία φορά από τους παραλήπτες που προορίζεται. Το QoS 2 είναι το ασφαλέστερο και πιο αργό επίπεδο ποιότητας υπηρεσιών. Η εγγύηση παρέχεται από τουλάχιστον δύο μηνύματα «request/response» μεταξύ του αποστολέα και του παραλήπτη.

4 παρουσίαση Συστήματος

Στο συγκεκριμένο κεφάλαιο θα αναφερθούμε στα υλικά μέρη του συστήματος και στον τρόπο λειτουργίας τους. Σαν βασικό στοιχείο έχουμε τον μικροελεκτή ESP32.

4.1 ESP32

Το ESP32 είναι ένας μικροελεγκτής (MCU) που σχεδιάστηκε από την Espressif Systems. Η Espressif Systems είναι μια κινεζική εταιρεία που έχει την έδρα της στην Shanghai. Υπάρχουν πολλές κατηγορίες υλοποίησης του συγκεκριμένου chip όπως ESP32-WROOM Series, ESP32-SOLO Series και ESP32-WROVER Series. Για την υλοποίηση του συστήματος της εργασίας χρησιμοποιήθηκαν πλακέτες που έχουν τον ESP32-WROOM-32.

Ο ESP32-WROOM-32 είναι μια ισχυρή, γενική μονάδα, η οποία διαθέτει Wi-Fi, Bluetooth και Bluetooth Low Energy (BLE). Στοχεύει μια μεγάλη ποικιλία εφαρμογών, από δίκτυα αισθητήρων χαμηλής ισχύος έως τις πιο απαιτητικές εργασίες, όπως κωδικοποίηση φωνής, ροή μουσικής και αποκωδικοποίηση MP3 . [15] [16]

Στον πυρήνα της συγκεκριμένης μονάδας βρίσκεται το chip ESP32-D0WDQ6. Το ενσωματωμένο chip έχει σχεδιαστεί ώστε να είναι επεκτάσιμο και προσαρμοστικό. Διαθέτει δύο πυρήνες CPU που μπορούν να ελεγχθούν ξεχωριστά και η συχνότητα του ρολογιού τους είναι ρυθμιζόμενη από 80 MHz έως 240 MHz. Το chip έχει επίσης έναν συν-επεξεργαστή χαμηλής ισχύος που μπορεί να χρησιμοποιηθεί αντί της CPU για την εξοικονόμηση ενέργειας κατά την εκτέλεση εργασιών που δεν απαιτούν πολύ υπολογιστική ισχύ, όπως την παρακολούθηση περιφερειακών αισθητήρων για αλλαγές.

Το ESP32 ενσωματώνει ένα πλούσιο σύνολο περιφερειακών χαρακτηριστικών, που κυμαίνονται από Capacitive Touch Sensors, Hall Sensors, διεπαφή SD card, Ethernet, SPI υψηλής ταχύτητας, UART, I2S και I2C.

Συγκεκριμένα το ESP32 διαθέτει τα εξής περιφερειακά χαρακτηριστικά:

- 18 Analog-to-Digital Converter (ADC) channels
- 10 Capacitive sensing GPIOs (Touch Sensors)
- 3 UART interfaces
- 3 SPI interfaces

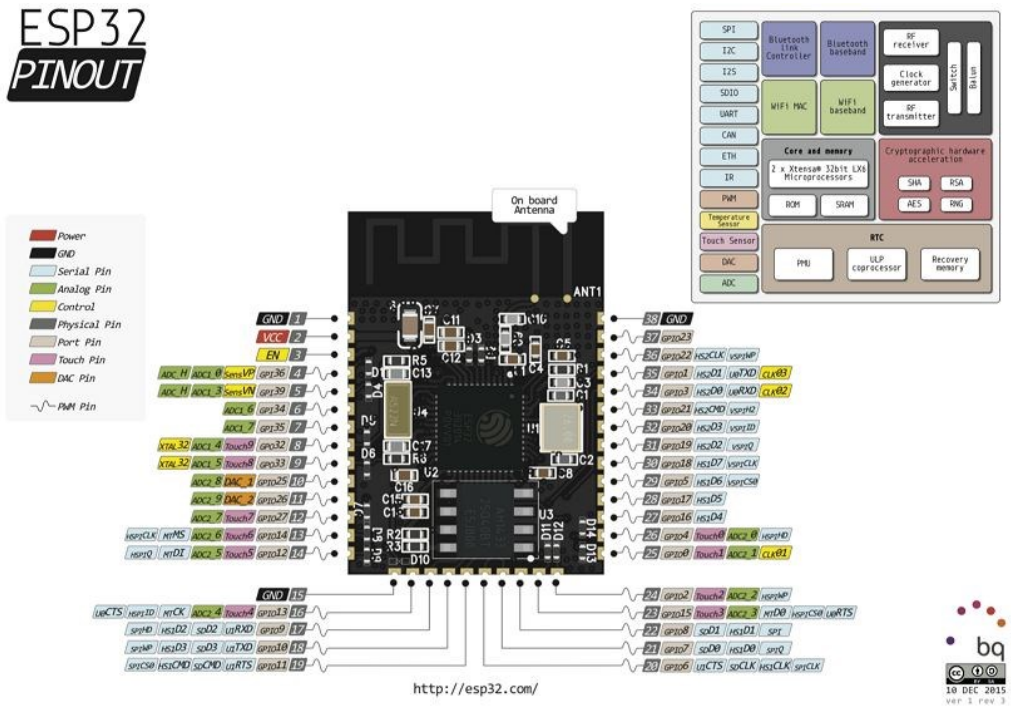
- 2 I2C interfaces
- 16 PWM output channels
- 2 Digital-to-Analog Converters (DAC)
- 2 I2S interfaces

Το ESP32-WROOM-32 έχει 38 ακίδες GPIO με πολλαπλές λειτουργίες, δηλαδή μια ακίδα GPIO μπορεί να συγκεντρώνει την λειτουργικότητα πολλών ιδιοτήτων (περιφερειακών χαρακτηριστικών).

Από αυτές οι GPIO6, GPIO7, GPIO8, GPIO9, GPIO10 και GPIO11 είναι συνδεδεμένες στο ενσωματωμένο flash SPI, ενσωματωμένο στη μονάδα και δεν συνιστώνται για άλλες χρήσεις. Δηλαδή δεν πρέπει να χρησιμοποιούνται για άλλους σκοπούς, είναι εκτός ορίων.

Στο ESP32 υπάρχουν δύο κατηγορίες αναλογικοί σε ψηφιακοί μετατροπείς (Analog-to-Digital Converter) που ονομάζονται ADC1 και ADC2. Το ADC1 έχει 8 κανάλια εισόδου, ενώ το ADC2 έχει 10 κανάλια εισόδου. Είναι σημαντικό να σημειωθεί ότι δεν μπορεί να χρησιμοποιηθεί το ADC2 ενώ είναι ενεργοποιημένο το WiFi, καθώς υπάρχει περίπτωση να εμφανιστούν προβλήματα στην απόκτηση της τιμής. [16][17]

Στην επόμενη εικόνα φαίνεται το pinout για το chip ESP32-WROOM-32.



Εικόνα 4 Pinout του ESP32-WROOM-32 (Πηγή: esp32, https://esp32.com/)

Εξαιτίας όλων αυτών των χαρακτηριστικών και της χαμηλής τιμής του, το ESP32 έχει

αποκτήσει τεράστια δημοτικότητα. Η ενσωμάτωση Bluetooth, Bluetooth LE και Wi-Fi είναι πολύ χρήσιμα εργαλεία και διασφαλίζουν ένα μεγάλο φάσμα εφαρμογών. Η ύπαρξη του συν-επεξεργαστή χαμηλής ισχύος και η ύπαρξη των ρυθμίσεων sleep modes, καθιστούν το ESP32 κατάλληλο για εφαρμογές με μπαταρία και φορητές ηλεκτρονικές κατασκευές.

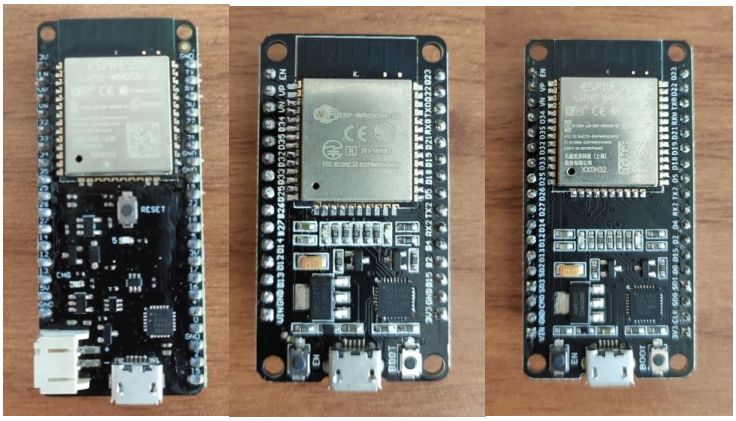
Αξίζει να τονιστεί ότι τα chip των ESP32 παράγονται από το εργοστάσιο σε μια ακατέργαστη μορφή. Για τους περισσότερους χρήστες η συγκεκριμένη μορφή δεν είναι εύκολα αξιοποιήσιμη. Το chip είναι πολύ μικρό και παρουσιάζει δυσκολία στο να συνδέσουμε καλώδια και περιφερειακές συσκευές, επειδή οι αποστάσεις των ακίδων είναι μικρές. Η μορφή του ESP32 απευθείας από το εργοστάσιο παραγωγής φαίνεται στην επόμενη εικόνα.

Εικόνα 5 Chip ESP32-WROOM-32 (Πηγή: digikey, https://www.digikey.lv/en/products/detail/espressif-systems/ESP32-WROOM-32-8MB/9381728)

Ωστόσο πολλές τρίτες OEM εταιρίες κατασκευάζουν τις δικές τους πλακέτες χρησιμοποιώντας τον ESP32. Οι συγκεκριμένες πλακέτες έχουν μια πιο φιλική προς τον χρήστη μορφή. Δίνονται στις ακίδες αποστάσεις κατάλληλες ώστε να μπορούν να συνδεθούν σε κάποιο breadboard. Κάποια από τα χαρακτηριστικά που δίνονται στις συγκεκριμένες πλακέτες είναι η τοποθέτηση ρυθμιστών τάσεων LDO ώστε να είναι πιο εύκολη η τροφοδοσία τους, θύρες micro USB ώστε να μπορούν να συνδέονται με τον υπολογιστή, reset button για να μπορεί να γίνει επανεκκίνηση, δυνατότητα σύνδεσης με μπαταρίες και πολλά άλλα χαρακτηριστικά. Αυτό έχει ως αποτέλεσμα να υπάρχει μια μεγάλη ποικιλία από πλακέτες που μπορούν να ικανοποιήσουν οποιαδήποτε ανάγκη και λειτουργικότητα επιθυμεί ο χρήστης.

Παρόλα αυτά χρειάζεται προσοχή στην επιλογή της επιθυμητής πλακέτας κυρίως όσο αφορά τις διαθέσιμες ακίδες GPIO που προσφέρει ο κάθε κατασκευαστής. Αν και οι ακίδες GPIO είναι καθορισμένες από το chip ESP32 και λειτουργούν με τον ίδιο τρόπο για οποιαδήποτε πλακέτα, ωστόσο η θέση που βρίσκονται πάνω στις OEM υλοποιήσεις μπορεί να διαφέρει από εταιρία σε εταιρία ή να μην δίνονται προς λειτουργία όλες οι ακίδες του chip. Για παράδειγμα μια εταιρία μπορεί να έχει την ακίδα GPIO 33 στην πάνω αριστερή μεριά και μια άλλη εταιρία να την έχει στην κάτω αριστερή μεριά, παρόλα αυτά και οι δύο πλακέτες αναφέρονται στην ακίδα 33 του chip ESP32, με αποτέλεσμα ο τρόπος που επικαλούνται να είναι ο ίδιος.

Στο έξυπνο σύστημα ελέγχου απόδοσης κεντρικής θέρμανσης κτηρίων χρησιμοποιούνται 3 διαφορετικά είδη πλακετών ESP32 από δύο διαφορετικές εταιρίες. Θα χρησιμοποιηθούν τέσσερα wemos lolin32, ένα Doit ESP32 devkit v1 με 30 ακίδες και ένα Doit ESP32 devkit v1 με 36 ακίδες.



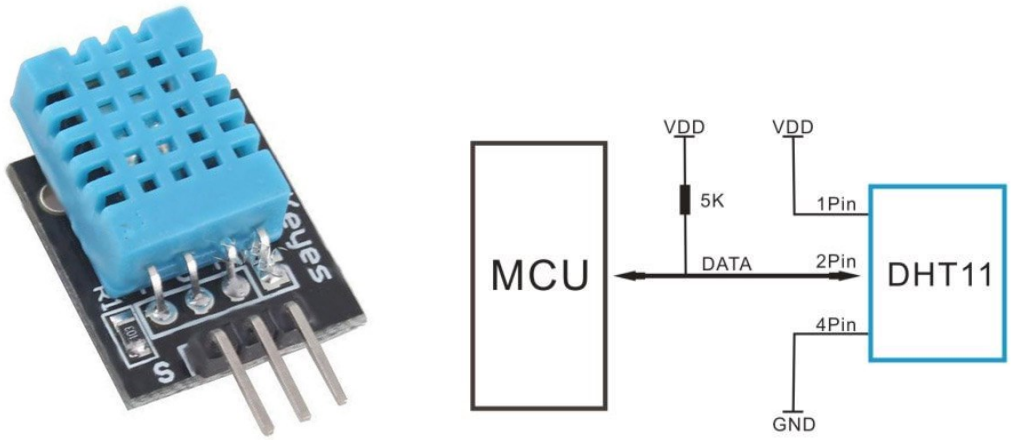
Εικόνα 6 Πλακέτες Lolin32, Doit 30pins και Doit 36pins αντίστοιχα

4.2 Αισθητήρες

Έχοντας επιλέγει τις επεξεργάστηκες μονάδες της υλοποίησης του συστήματος το επόμενο κομμάτι της κατασκευής είναι η επιλογή των αισθητήρων που θα λαμβάνουν τις τιμές και θα συνδέονται με τους μικροελεγκτές. Συνολικά θα χρησιμοποιηθούν 4 αισθητήρες υγρασίας και θερμοκρασίας με την ονομασία «DHT11», 3 αισθητήρες θερμοκρασίας επαφής με την ονομασία «DS18B20», 1 αισθητήρας φωτισμού με την ονομασία «BH1750» και 1 relay 4 καναλιών.

4.2.1 Αισθητήρας Θερμοκρασίας Και Υγρασίας DHT11

Για την μέτρηση της θερμοκρασία και της υγρασία του χώρου χρησιμοποιήθηκε ο αισθητήρας DHT11. Ο DHT11 μπορεί να μετρήσει θερμοκρασία από 0 °C έως 50 °C με ακρίβεια ± 2,0 °C και υγρασία από 20 έως 80% με ακρίβεια 5%. Η τάση τροφοδοσίας του κυμαίνεται από 3,3V έως 5,5V. Σε περίπτωση τροφοδοσίας με 5V, ο αισθητήρας μπορεί να παραμείνει έως και 20 μέτρα μακριά από την πλακέτα. Ωστόσο, με τάση τροφοδοσίας 3,3V, το μήκος του καλωδίου δεν πρέπει να είναι μεγαλύτερο από 1 μέτρο. Οι αισθητήρες DHT11 απαιτούν συνήθως εξωτερική αντίσταση 5Kohm μεταξύ του VCC και του pin μεταφοράς δεδομένων προκειμένου να μπορεί να επιτευχθεί η σωστή επικοινωνία μεταξύ του αισθητήρα και της πλακέτας. Ωστόσο, στην αγορά υπάρχουν και μονάδες με ενσωματωμένη την απαιτούμενη αντίσταση χωρίς μεγάλη διαφορά στην τιμή. Για την συγκεκριμένη εργασία, προμηθεύτηκαν τέτοιας κατηγορίας μονάδες.



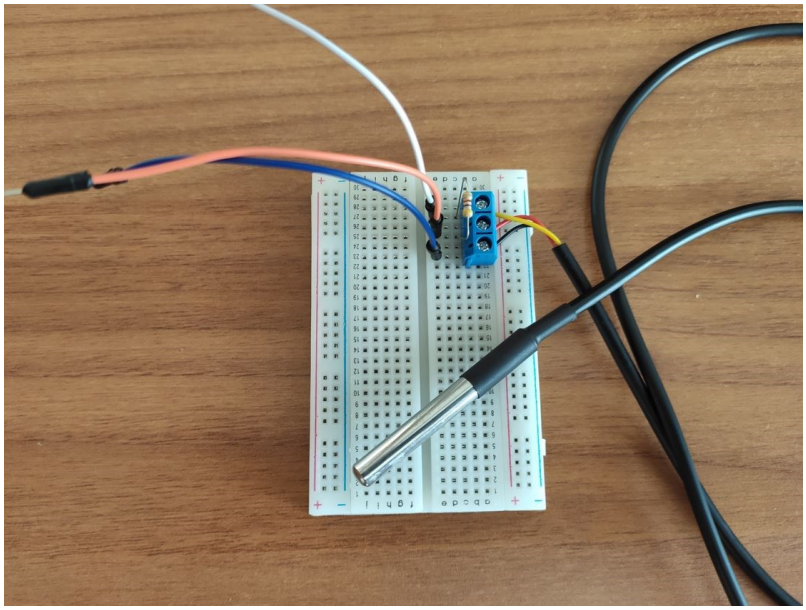
Εικόνα 7 Αισθητήρας DHT11 (Πηγή: components101, https://components101.com/sensors/dht11-temperature-sensor)

Από τα χαρακτηριστικά του DHT11 γίνεται κατανοητό ότι δεν είναι ιδανικός για το σύστημα που υλοποιήθηκε, εξαιτίας του μικρού εύρους τιμών στην θερμοκρασία που διαθέτει. Ωστόσο επιλέχθηκε επειδή είναι ο οικονομικότερος της αγοράς με τιμή στα 1,6 ευρώ. Ένας πρόσθετος λόγος επιλογής είναι ότι ο κώδικας με τον οποίο υλοποιήθηκε μπορεί να χρησιμοποιηθεί και για τον DHT22 με εύρος θερμοκρασίας από -40 °C έως 80 °C και τιμή αγοράς τα 5,5 ευρώ. Οπότε με την διάθεση παραπάνω χρημάτων και χρησιμοποιώντας τα ίδια εργαλεία είναι δυνατόν να κατασκευαστεί ένα ακριβέστατο σύστημα.

4.2.2 Αισθητήρας Θερμοκρασίας DS18B20

Ο DS18B20 είναι αισθητήρας θερμοκρασίας που χρησιμοποιεί την τεχνολογία 1-Wire και κατασκευάζεται από την Dallas Semiconductor Corp. Ένα από τα βασικά πλεονεκτήματα του συγκεκριμένου αισθητήρα είναι ότι έχει μεγάλο εύρος μέτρησης της θερμοκρασίας από -55 °C έως +125°C με ακρίβεια ± 0,5 °C. Επίσης διαθέτει δύο εκδόσεις μία απλή στην οποία θυμίζει τρανζίστορ και μία αδιάβροχη η οποία είναι χρήσιμη όταν απαιτούνται μετρήσεις με επαφή. Για την συγκεκριμένη εργασία χρησιμοποιήθηκε η αδιάβροχη έκδοση επειδή χρησιμοποιήθηκε με επαφή πάνω στα θερμαντικά σώματα και στους σωλήνες μεταφορά νερού.

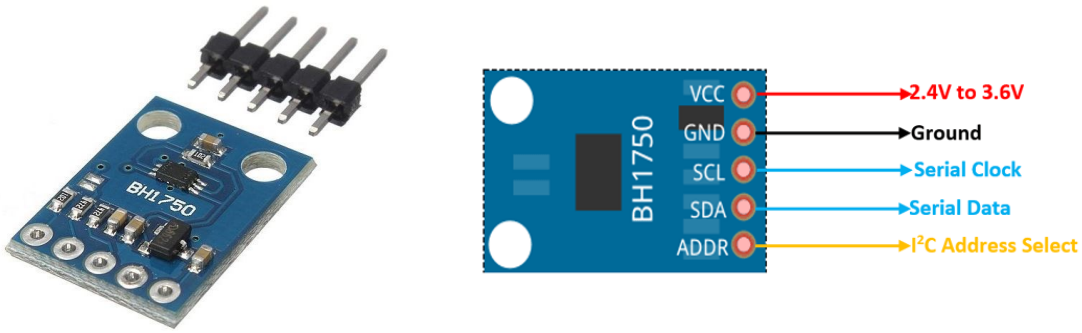
Όπως και ο DHT11 έτσι και ο αισθητήρας DS18B20 απαιτεί μια εξωτερική αντίσταση 4,7Kohm μεταξύ του VCC και του pin μεταφοράς δεδομένων προκειμένου να μπορεί να επιτευχθεί η σωστή επικοινωνία μεταξύ του αισθητήρα και της πλακέτας. Όμως στην συγκεκριμένη περίπτωση οι μονάδες με την ενσωματωμένη αντίσταση κοστίζουν την διπλάσια τιμή από αυτές που δεν την περιέχουν, οπότε προτιμήθηκε να γίνει το κύκλωμα από εμάς. Η συνδεσμολογία που υλοποιήθηκε φαίνεται στην επόμενη φωτογραφία.



Εικόνα 8 Αισθητήρας DS18B20

4.2.3 Αισθητήρας Φωτεινότητας BH1750

Ο BH1750 είναι ένας ψηφιακός αισθητήρας φωτισμού και μπορεί να μετρήσει με υψηλή ανάλυση και μεγάλη εμβέλεια την ισχύ του φωτός (LUX) από 1 lx έως 65535 lx. Η μονάδα BH1750 επικοινωνεί μέσω του διαύλου I2C με τον ελεγκτή ESP32 για τη μετάδοση των μετρημένων δεδομένων και λειτουργεί σε εύρος τάσης 2,4 V - 3,6 V.



Εικόνα 9 Αισθητήρας BH1750 (Πηγή: components101, https://components101.com/sensors/bh1750-ambient-light-sensor)

Ένας αισθητήρας φωτεινότητας μετρά την ένταση του ηλιακού φωτός, όμως ο στόχος της συγκεκριμένης εργασίας είναι να μετρηθεί η ηλιακή ακτινοβολία. Η άμεση μετατροπή του ηλιακού φωτός (lux) σε τιμή ηλιακής ακτινοβολίας (W/m2) δεν είναι δυνατή μιας και η φύση της μετρούμενης παραμέτρου είναι διαφορετική. Ωστόσο το κόστος ενός πυρανόμετρου είναι μεγάλο και ένα από τα χαρακτηριστικά που μας ενδιαφέρει είναι να κάνουμε όσο γίνεται πιο οικονομικό το σύστημα που υλοποιήσαμε. [18]

Αν και υπάρχει μεγάλο πλήθος βιβλιογραφίας και διαδικτυακών πηγών που με την χρήση οργάνων και συλλογή δεδομένων προσπαθούν να εντοπίσουν έναν συντελεστή μετατροπής της ηλιακής φωτεινότητας σε ηλιακή ακτινοβολία, πολλά από αυτά καταλήγουν σε αντικρουόμενα αποτελέσματα. Για την συγκεκριμένη εργασία χρησιμοποιήθηκε ο κανόνας ότι 122 lx ισούται με 1 W/m2, όπως συμβουλεύει το άρθρο «A conversion guide: solar irradiance and lux illuminance» των Peter R. Michael , Danvers E. Johnston και Wilfrido Moreno. [19]

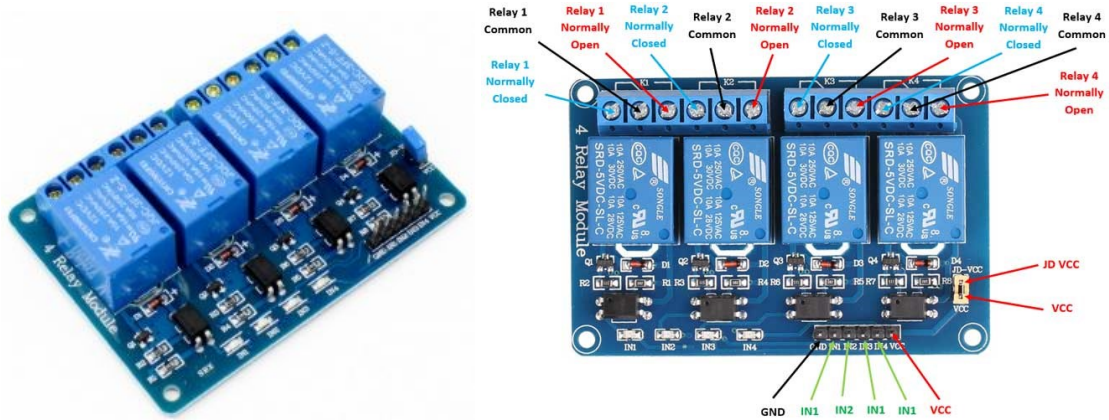
4.3 Relay

Τα relay (ηλεκτρονόμοι) είναι ηλεκτρικοί διακόπτες που ανοίγουν και κλείνουν ένα ηλεκτρικό κύκλωμα με εντολές ενός άλλου ηλεκτρικού κυκλώματος. Ο απλούστερος ηλεκτρονόμος λειτουργεί μέσω ενός ηλεκτρομαγνήτη που ενεργοποιεί τον διακόπτη με το άνοιγμα, κλείσιμο μιας ή περισσότερων επαφών. Για την συγκεκριμένη εργασία χρησιμοποιήθηκε μια μονάδα relay τεσσάρων καναλιών, δηλαδή περιέχει τέσσερα relay. Η συγκεκριμένη μονάδα έχει τάση τροφοδοσίας τα 5V και το κάθε relay έχει μέγιστη τάση επαφής τα 250 VAC και 30 VDC, και μέγιστο ρεύμα τα 10 A.

Προκειμένου να επιτευχθεί η επιθυμητή λειτουργικότητα του κυκλώματος πρέπει να δοθεί μεγάλη προσοχή στην συνδεσμολογία κατά την σύνδεση των relay με το κύκλωμα που θέλουμε να χειριζόμαστε. Κάθε μονάδα relay έχει τρεις υποδοχές: Common, Normally Closed και Normally Open. Οπότε ανάλογα με το ποιες επαφές χρησιμοποιούνται, προκείπτουν δύο είδη λειτουργικότητας:

1) Η Normally Open χρησιμοποιείται όταν απαιτείται το ηλεκτρικό κύκλωμα να είναι ανοικτό (δεν διαρρέεται με ρεύμα) όταν ο ηλεκτρονόμος απενεργοποιείται και κλειστό (διαρρέεται με ρεύμα) όταν ο ηλεκτρονόμος ενεργοποιείται. Για αυτή την λειτουργία χρησιμοποιούνται οι επαφές Common και Normally Open.

2) Η Normally Closed χρησιμοποιείται όταν απαιτείται το ηλεκτρικό κύκλωμα να είναι κλειστό (διαρρέεται με ρεύμα) όταν ο ηλεκτρονόμος απενεργοποιείται και ανοικτό (δεν διαρρέεται με ρεύμα) όταν ο ηλεκτρονόμος ενεργοποιείται. Για αυτή την λειτουργία χρησιμοποιούνται οι επαφές Common και Normally Closed.



Εικόνα 10 Relay (Πηγή: components101, https://components101.com/switches/5v-four-channel-relay-module-pinout-features-applications-working-datasheet)

4.4 Πρόσθετα Βοηθητικά Υλικά

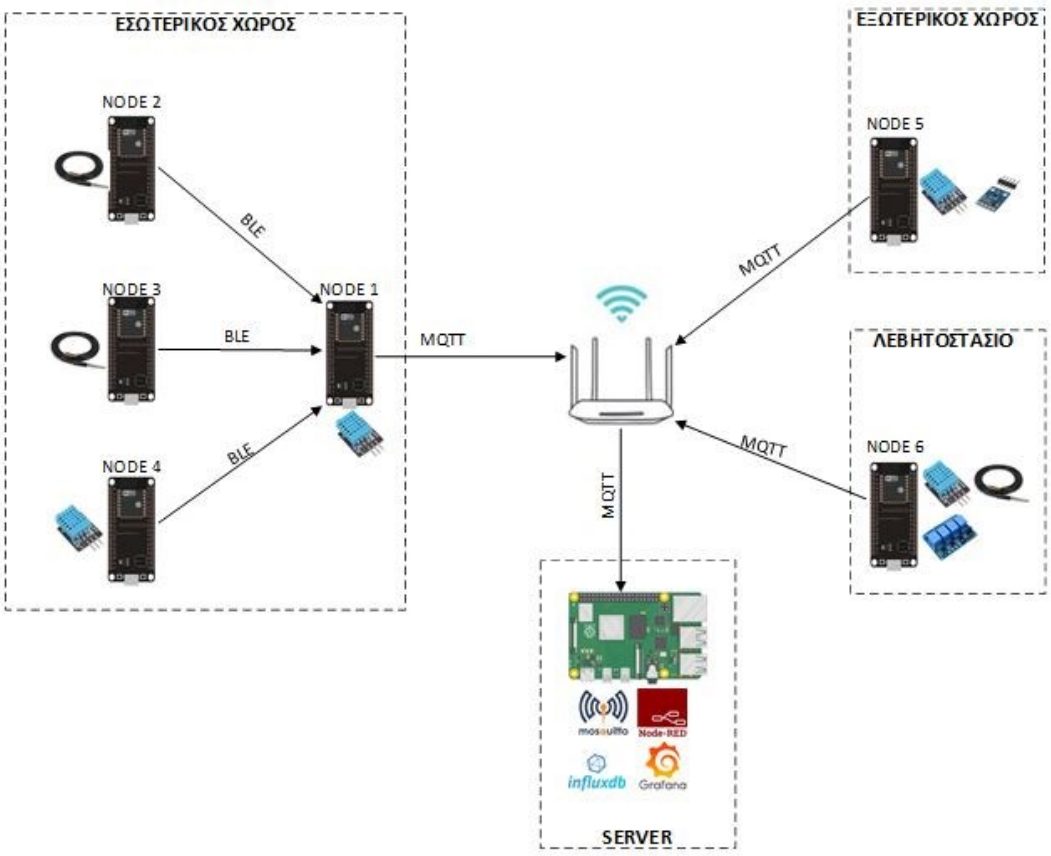
Τέλος για την υλοποίηση του συστήματος χρειάστηκαν κάποια πρόσθετα υλικά προκειμένου να επιτευχθεί η συνδεσμολογία και λειτουργία των διάφορων μονάδων. Τα πρόσθετα υλικά ήταν καλώδια, μικρές κλέμες για τους αισθητήρες DS18B20, breadboard, μπαταρίες 18650 των 3,7 V και αντιστάσεις.

Εικόνα 11 Πρόσθετα Υλικά

4.5 Σύστημα

Το συγκεκριμένο σύστημα χωρίζεται σε 4 υποσυστήματα και συνολικά διαθέτει 6 πλακέτες με επεξεργαστή ESP32 και μια πλακέτα raspberry pi 4, η οποία τελεί χρέη διακομιστή. Το σύστημα χωρίζεται στο υποσύστημα εσωτερικού χώρου, το υποσύστημα εξωτερικού χώρου, το υποσύστημα λεβητοστασίου και τον διακομιστή. Στο συγκεκριμένο κεφάλαιο θα αναλυθούν μόνο τα υποσυστήματα που υλοποιούν τον έξυπνο θερμοστάτη, ο διακομιστής θα αναλυθεί στο επόμενο κεφάλαιο.

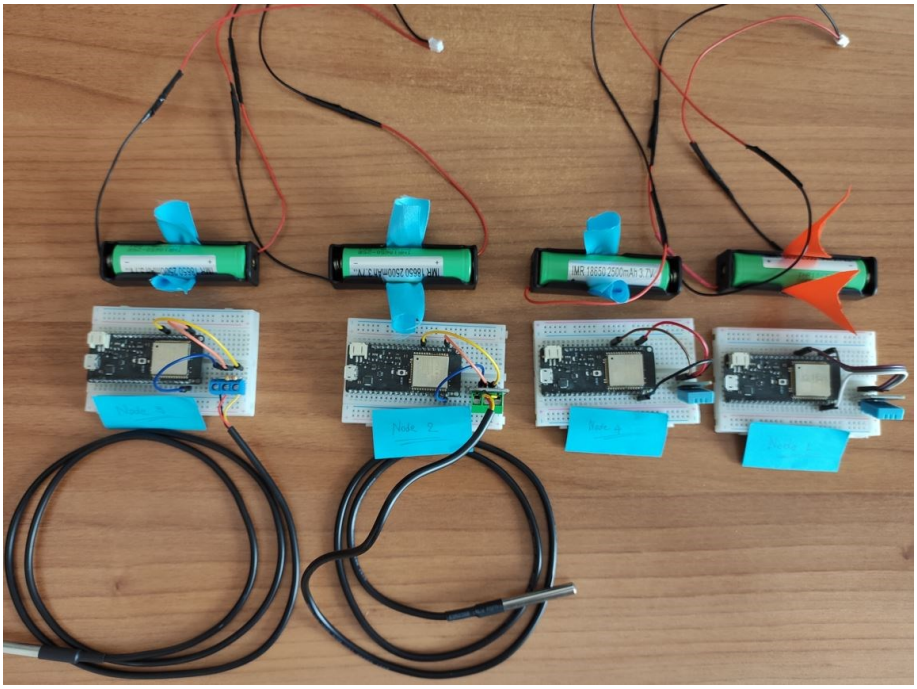
Βασική προϋπόθεση για την ομαλή λειτουργία του συγκεκριμένου συστήματος είναι ότι στον χώρο υπάρχει πρόσβαση σε δίκτυο WiFi και διασύνδεση στο διαδίκτυο.



Εικόνα 12 Τρόπος Επικοινωνίας Συστήματος

4.5.1 Υποσύστημα Εσωτερικού Χώρου

Το Υποσύστημα Εσωτερικού Χώρου αποτελείται από 4 πλακέτες lolin32, μία των οποίων τελεί χρέη gateway. Συνολικά διαθέτει 2 αισθητήρες DHT11 για την καταμέτρηση της θερμοκρασίας και υγρασίας του εσωτερικού χώρου και 2 αισθητήρες DS18B20 επαφής για την καταμέτρηση της θερμοκρασίας του θερμικού σώματος. Οι πλακέτες συνδέονται μεταξύ τους με το πρωτόκολλο επικοινωνίας Bluetooth Low Energy (BLE). Η πλακέτα που έχει οριστεί ως gateway συγκεντρώνει τα δεδομένα και τα αποστέλλει διαμέσου του WiFi και χρησιμοποιώντας το πρωτόκολλο MQTT στον Mosquitto MQTT Broker, ο οποίος φιλοξενείται από το raspberry pi 4.

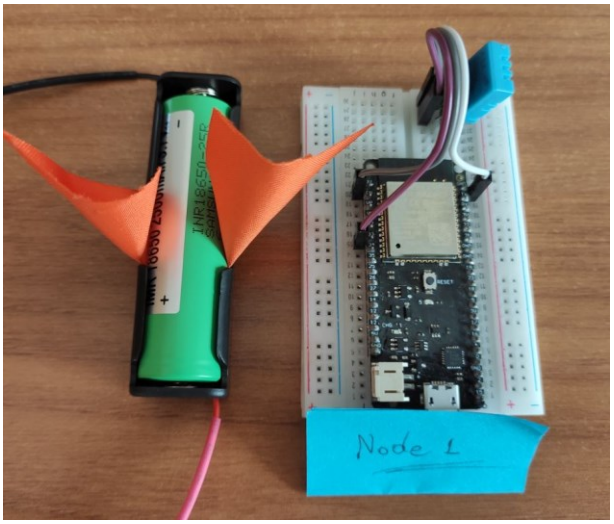


Εικόνα 13 Πλακέτες Εσωτερικού Χώρου

<u>Παρουσίαση των πλακετών που αποτελούν το υποσύστημα του εσωτερικού χώρου</u>:

1) **<u>Node 1</u>**: Node 1 ονομάζεται η κεντρική πλακέτα του υποσυστήματος εσωτερικού χώρου. Διαθέτει έναν αισθητήρα **DHT11** για να καταμετρά την θερμοκρασία και την υγρασία στον εσωτερικό χώρο του σπιτιού. Εκτός από τις προσωπικές του μετρήσεις, η συγκεκριμένη πλακέτα τελεί χρέη BLE gateway, συγκεντρώνει τις μετρήσεις από τις υπόλοιπες πλακέτες και τις στέλνει στον διακομιστή. Για τον λόγο αυτό πρέπει η συγκεκριμένη πλακέτα να τοποθετηθεί σε μέρος με καλό σήμα WiFi. Επίσης συνδέεται με μια μπαταρία 18650 προκειμένου να είναι εφικτή η ασύρματη λειτουργία του.

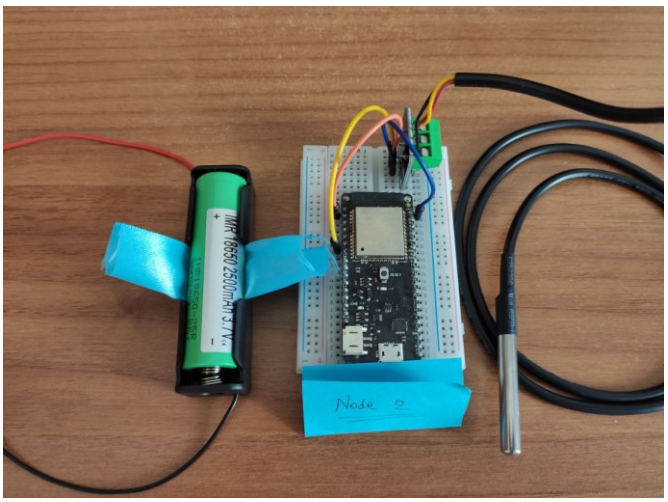
<u>Διαδικασία λειτουργίας της πλακέτας Node 1</u>: Αρχικά ενεργοποιούνται οι κεραίες BLE και WiFi της πλακέτας προκειμένου να είναι εφικτή η Bluetooth Low Energy και WiFi επικοινωνία. Ορίζεται η πλακέτα ως BLE Client και καθορίζεται ο MQTT Broker που θα σταλούν τα δεδομένα. Η πλακέτα χρησιμοποιώντας την λειτουργία BLE αναζητά στον γύρο χώρο τους BLE Server. Χρησιμοποιώντας έναν αισθητήρα DHT11 καταμετρά την θερμοκρασία και την υγρασία της περιοχής που έχει τοποθετηθεί. Αργότερα στέλνει request στους εντοπισμένους BLE Server προκειμένου να της σταλούν οι μετρήσεις τους. Τέλος στέλνει διαμέσου πρωτοκόλλου MQTT τόσο τις δικιές της μετρήσεις όσο και τις μετρήσεις των υπόλοιπων nodes στον Mosquitto Broker.



Εικόνα 14 Node 1 gateway

2) **Node 2**: Node 2 ονομάζεται η πλακέτα που μετρά την θερμοκρασία, στο πάνω μέρος του θερμαντικού σώματος. Διαθέτει έναν αισθητήρα **DS18B20** waterproof, ο οποίος τοποθετείται με επαφή πάνω στο θερμαντικό σώμα. Επίσης σύνδέεται με μια μπαταρία 18650 προκειμένου να είναι εφικτή η ασύρματη λειτουργία του.

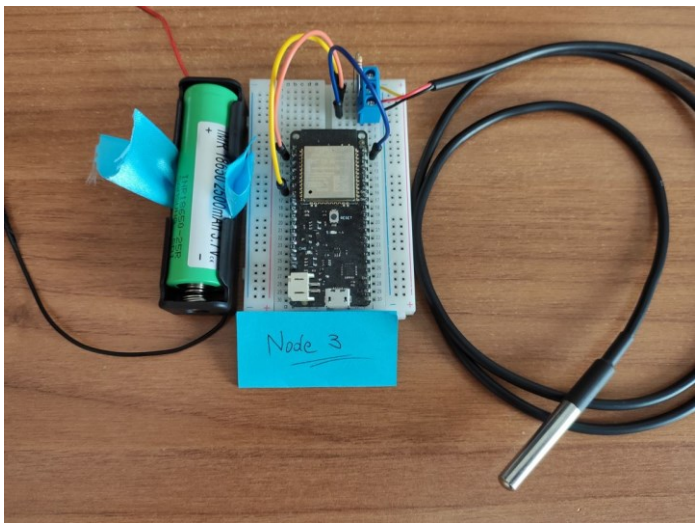
Διαδικασία λειτουργίας της πλακέτας Node 2: Αρχικά ενεργοποιείται η BLE κεραία της πλακέτας προκειμένου να είναι εφικτή η Bluetooth Low Energy επικοινωνία και την ορίζουμε ως BLE Server. Η πλακέτα μετρά την θερμοκρασία, χρησιμοποιώντας τον DS18B20 από το θερμαντικό σώμα και διαφημίζει την ύπαρξή της στον γύρο χώρο. Όταν της ζητηθεί από το gateway η θερμοκρασία την αποστέλλει διαμέσου του πρωτοκόλλου BLE.



Εικόνα 15 Node 2

3) **Node 3**: Node 3 ονομάζεται η πλακέτα που μετρά την θερμοκρασία, στο σωλήνα ροής νερού του θερμαντικού σώματος. Διαθέτει έναν αισθητήρα **DS18B20** waterproof, ο οποίος τοποθετείται με επαφή στον σωλήνα. Επίσης συνδέεται με μια μπαταρία 18650 προκειμένου να είναι εφικτή η ασύρματη λειτουργία του.

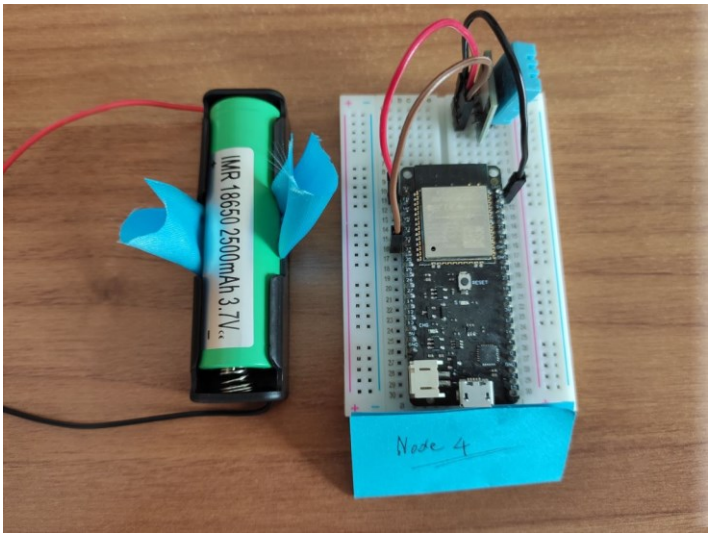
Διαδικασία λειτουργίας της πλακέτας Node 3: Αρχικά ενεργοποιείται η BLE κεραία της πλακέτας προκειμένου να είναι εφικτή η Bluetooth Low Energy επικοινωνία και την ορίζουμε ως BLE Server. Η πλακέτα μετρά την θερμοκρασία, χρησιμοποιώντας τον DS18B20, από τον σωλήνα και διαφημίζει την ύπαρξή της στον γύρο χώρο. Όταν της ζητηθεί από το gateway η θερμοκρασία τότε την αποστέλλει διαμέσου του πρωτοκόλλου BLE.



Εικόνα 16 Node 3

3) **Node 4**: Node 4 ονομάζεται η πλακέτα που μετρά την θερμοκρασία και υγρασία, στον εσωτερικό χώρο του σπιτιού. Διαθέτει έναν αισθητήρα **DHT11** και έχει τοποθετηθεί σε διαφορετική περιοχή του χώρου από ότι το Node 1. Επίσης έχει συνδεθεί με μια μπαταρία 18650 προκειμένου να είναι εφικτή η ασύρματη λειτουργία του.

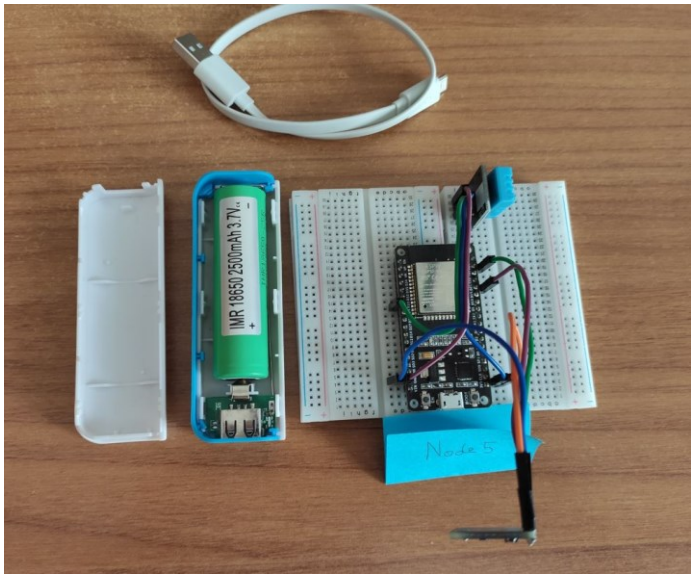
Διαδικασία λειτουργίας της πλακέτας Node 4: Αρχικά ενεργοποιείται η BLE κεραία της πλακέτας προκειμένου να είναι εφικτή η Bluetooth Low Energy επικοινωνία και την ορίζουμε ως BLE Server. Η πλακέτα μετρά την θερμοκρασία και την υγρασία, χρησιμοποιώντας τον DHT11 και διαφημίζει την ύπαρξή της στον γύρο χώρο. Όταν της ζητηθούν από το gateway οι τιμές του αισθητήρα τότε τις αποστέλλει διαμέσου του πρωτοκόλλου BLE.



Εικόνα 17 Node 4

4.5.2 Υποσύστημα Εξωτερικού Χώρου

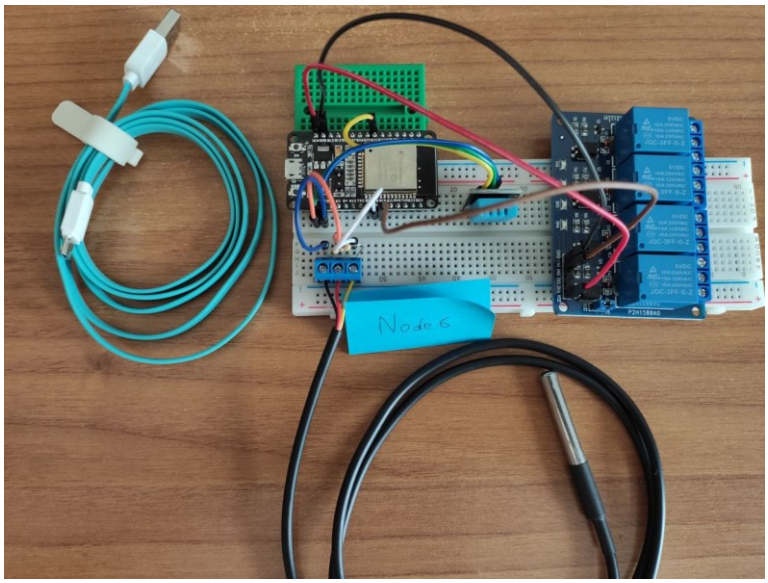
Το Υποσύστημα Εξωτερικού Χώρου αποτελείται από μια πλακέτα ESP32 και συγκεκριμένα την DOIT ESP32 DEVKIT V1 με 36 pins. Η συγκεκριμένη πλακέτα είναι συνδεδεμένη με έναν αισθητήρα **DHT11** για την καταμέτρηση της θερμοκρασίας και της υγρασίας του εξωτερικού χώρου. Επίσης διαθέτει τον αισθητήρα φωτεινότητας **BH1750** προκειμένου να καταμετρά την ηλιακή ακτινοβολία. Συνδέεται στο τοπικό δίκτυο με την κεραία WiFi και διαμέσου του πρωτοκόλλου MQTT μεταδίδει τα δεδομένα που συλλέγει στον MQTT Broker Mosquitto. Ως σύστημα τροφοδοσίας η συγκεκριμένη πλακέτα χρησιμοποιεί ένα powerbank το οποίο διαθέτει μια μπαταρία 18650 και συνδέεται στην πλακέτα διαμέσου του micro USB.



Εικόνα 18 Node 5

4.5.3 Υποσύστημα Λεβητοστασίου

Το Υποσύστημα Λεβητοστασίου αποτελείται από μια πλακέτα ESP32 και συγκεκριμένα την DOIT ESP32 DEVKIT V1 με 30 pins. Η συγκεκριμένη πλακέτα είναι συνδεδεμένη με έναν αισθητήρα **DHT11** για την καταμέτρηση της θερμοκρασίας και της υγρασίας του υπογείου, όπου είναι τοποθετημένος ο λέβητας. Διαθέτει έναν πρόσθετο αισθητήρα θερμοκρασίας **DS18B20** waterproof, ο οποίος διαμέσου επαφής καταμετρά την θερμοκρασία του σωλήνα, για να γνωρίζουμε την θερμοκρασία που ξεκινά το νερό από τον κυκλοφορητή. Για να καταστεί δυνατόν να ελέγχεται η λειτουργία του λέβητα προστέθηκε ένα relay προκειμένου να είναι δυνατή η ενεργοποίηση και απενεργοποίηση του διαμέσου της διεπαφής χρήστη που θα δημιουργηθεί με το Node-Red. Για την τροφοδοσία της χρησιμοποιείται ένα καλώδιο micro USB που τροφοδοτείται με ρεύμα από έναν φορτηστή κινητού.



Εικόνα 19 Node 6

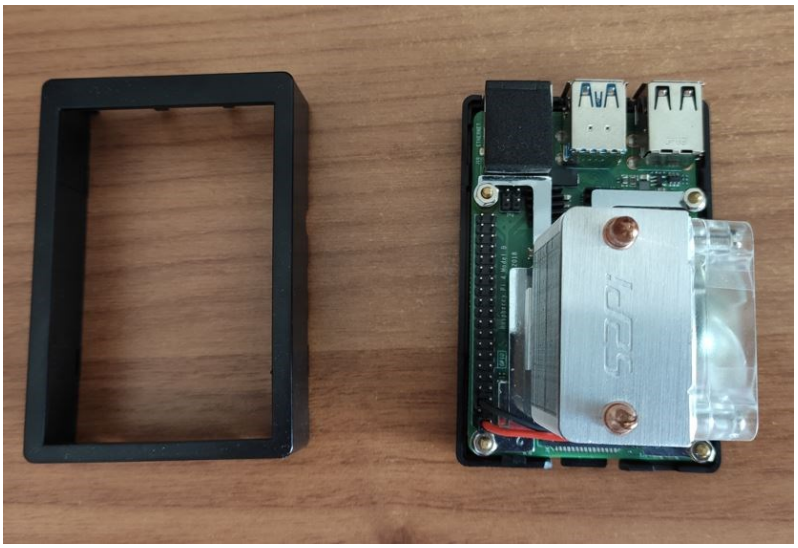
Ο κώδικας όλων των υλοποιήσεων έχει παρατεθεί στο Παράρτημα Α στο τέλος της εργασίας.

5 Συλλογή Και Επεξεργασία Δεδομένων Συστήματος

Στο συγκεκριμένο κεφάλαιο, θα παρουσιαστεί το υλικό και το λογισμικό που θα υλοποιούν το υποσύστημα του εξυπηρετητή. Το συγκεκριμένο υποσύστημα είναι υπεύθυνο για την αποθήκευση και την παρουσίαση των δεδομένων του συστήματος που υλοποιείται στην εργασίας.

5.1 Raspberry Pi 4

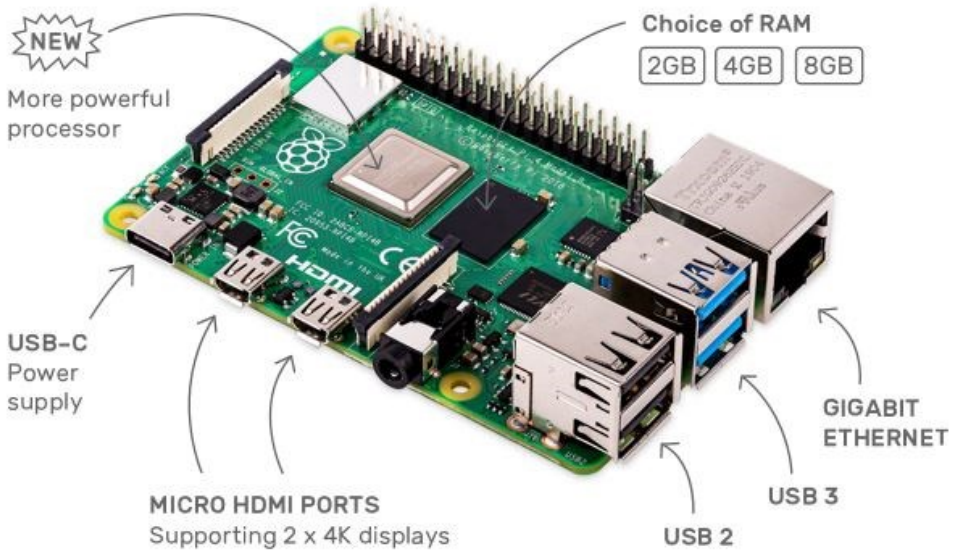
Ως εξυπηρετητής θα χρησιμοποιηθεί μια πλακέτα Raspberry Pi 4 Model B 4GB Ram, η οποία θα συνδέεται στο τοπικό δίκτυο του δρομολογητή διαμέσου ενός ethernet καλωδίου και περιέχει μια κάρτα microSD 64GB στην οποία είναι εγκατεστημένα όλα τα απαραίτητα λογισμικά.



Εικόνα 20 Raspberry Pi 4 Συστήματος

Το Raspberry Pi είναι μια σειρά μικρών υπολογιστών μονής πλακέτας (SBC) που αναπτύχθηκαν στο Ηνωμένο Βασίλειο από το Ίδρυμα Raspberry Pi σε συνεργασία με την Broadcom. Αν και αρχικά ο σκοπός της δημιουργίας της πρώτης πλακέτας ήταν για εκπαιδευτικούς σκοπούς, δεν άργησε να γίνει πιο δημοφιλές από το αναμενόμενο κάνοντας πωλήσεις εκτός της αγοράς στόχου του. Η μεγάλη ζήτηση των Raspberry Pi οφείλεται κυρίως σε χομπίστες ηλεκτρονικών κατασκευών εξαιτίας του χαμηλού κόστους, της χαμηλής κατανάλωσης ενέργειας και του ανοικτού σχεδιασμού τους. [20]

Το Raspberry Pi 4 Model B είναι το τελευταίο μοντέλο της σειράς υπολογιστών Raspberry Pi. Το Raspberry Pi 4 είναι διαθέσιμο σε 3 εκδόσεις με διαφορετική μνήμη Ram, με 2GB, με 4GB και με 8GB.



Εικόνα 21 Ports Raspberry Pi 4 (Πηγή: raspberrypi, https://www.raspberrypi.org/products/raspberry-pi-4-model-b/)

Το Raspberry Pi 4 Model B παρέχει επιδόσεις ενός κλασικού συστήματος Η/Υ, μπορεί να συνδεθεί με δύο οθόνες, πληκτρολόγιο και ποντίκι. Ο χρήστης μπορεί να αποκωδικοποιεί βίντεο 4K, να έχει ταχεία αποθήκευση σε εξωτερικά μέσα με τις USB 3.0 θύρες και ταχύτερες συνδέσεις δικτύου μέσω του Gigabit Ethernet.

Στην παρακάτω λίστα φαίνονται τα χαρακτηριστικά του Raspberry Pi 4 Model B που χρησιμοποιείται ως διακομιστής στο σύστημα: [21]

Επεξεργαστής :	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
Μνήμη :	4GB LPDDR4-3200 SDRAM
Συνδεσιμότητα :	2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE 2 USB 3.0 ports; 2 USB 2.0 ports. Gigabit Ethernet

GPIO :	Raspberry Pi standard 40 pin GPIO header (πλήρως συμβατή με προηγούμενες πλακέτες)
Video & sound :	2 × micro-HDMI ports (υποστηρίζεται έως και 4kp60)
	2-lane MIPI DSI display port
	2-lane MIPI CSI camera port
	4-pole stereo audio and composite video port
Multimedia :	H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode)
	OpenGL ES 3.1, Vulkan 1.0
SD card support :	Υποδοχή κάρτας Micro-SD για φόρτωση λειτουργικού συστήματος και αποθήκευσης δεδομένων
Input power :	5V DC via USB-C connector (minimum 3A*)
	5V DC via GPIO header (minimum 3A*)
	Power over Ethernet (PoE) enabled (απαιτεί ξεχωριστό PoE HAT)
Environment :	Θερμοκρασία λειτουργίας : 0 - 50 βαθμοί C περιβάλλοντος

5.2 Raspberry Pi OS

Πολλά λειτουργικά συστήματα είναι διαθέσιμα για το Raspberry Pi, όπως το Raspberry Pi OS, OpenELEC, Windows IoT Core, RetroPie, Ubuntu Core, Ubuntu Mateto κ.α. Όπως αναφέρθηκε στην προηγούμενη ενότητα το Raspberry Pi είναι μια πλακέτα με μεγάλο πλήθος δυνατοτήτων, μπορεί να χρησιμοποιηθεί ως Server, ως παιχνιδομηχανή, ως TV box, ως Η/Υ κ.α. Ανάλογα με την λειτουργικότητα που επιθυμεί ο χρήστης να δώσει στην πλακέτα, επιλέγει το αντίστοιχο λειτουργικό.

Στην συγκεκριμένη περίπτωση εγκαταστάθηκε το Raspberry Pi OS (παλιότερα γνωστό ως Raspbian), που είναι το επίσημο υποστηριζόμενο λειτουργικό σύστημα για τα Raspberry Pi. Το Raspberry Pi OS είναι λογισμικό ανοιχτού κώδικα και είναι βασισμένο στο Debian, βελτιστοποιημένο για τις πλακέτες Raspberry Pi.

Η εγκατάσταση του λειτουργικού συστηματος μπορεί να γίνει εύκολα και γρήγορα σε μια κάρτα microSD διαμέσου του λογισμικού Raspberry Pi Imager, το οποίο μπορεί να βρεθεί στην επίσημη ιστοσελίδα της εταιρίας.[22]

https://www.raspberrypi.org/software/

5.3 Eclipse Mosquitto MQTT Broker

Προκειμένου να μπορεί το Raspberry Pi να υποστηρίζει το πρωτόκολλο MQTT, εγκαταστάθηκε το λογισμικό διακομιστή που ονομάζεται Eclipse Mosquitto MQTT Broker. Το Eclipse Mosquitto είναι ένας μεσίτης ανοιχτού κώδικα (με άδεια EPL / EDL) που εφαρμόζει το πρωτόκολλο MQTT εκδόσεις 5.0, 3.1.1 και 3.1.

Το Mosquitto είναι ελαφρύ λογισμικό και είναι κατάλληλο για χρήση σε όλες τις συσκευές από πλήρεις διακομιστές έως υπολογιστές μονής πλακέτας χαμηλής ισχύος, σαν το raspberry pi. Συνήθως, η τρέχουσα εφαρμογή του Mosquitto καταναλώνει περίπου 3MB RAM με 1000 πελάτες συνδεδεμένους. Επίσης παρέχει μια βιβλιοθήκη C για την εφαρμογή MQTT clients, και την γραμμή εντολών mosquitto_pub και mosquitto_sub. Το Mosquitto είναι μέρος του Eclipse Foundation και χρηματοδοτείται από το cedalo.com. [23]

Για την εγκατάσταση του Eclipse Mosquitto MQTT εκτελούμε στο παράθυρο εντολών του raspberry pi τις εντολές:

sudo apt update
sudo apt install -y mosquitto mosquitto-clients

Προκειμένου να ενεργοποιείται ο broker αυτόματα με την ενεργοποίηση ή επανεκκίνηση του raspberry pi γράφουμε την εντολή:

sudo systemctl enable mosquitto.service

5.4 Node-Red

Το Node-Red είναι ένα ισχυρό εργαλείο ανοικτού κώδικα για την δημιουργία εφαρμογών Internet of Things (IoT) με στόχο την απλοποίηση του στοιχείου του προγραμματισμού. Το ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) του Node-Red αρχικά αναπτύχθηκε από την IBM το 2013. Χρησιμοποιεί οπτικό προγραμματισμό και ο προγραμματιστής καλείται να

συνδέσει μεταξύ τους προκαθορισμένα μπλοκ κώδικα προκειμένου να εκτελεστεί μια διεργασία. Αυτά τα μπλοκ κώδικα ονομάζονται κόμβοι ή nodes. [24]

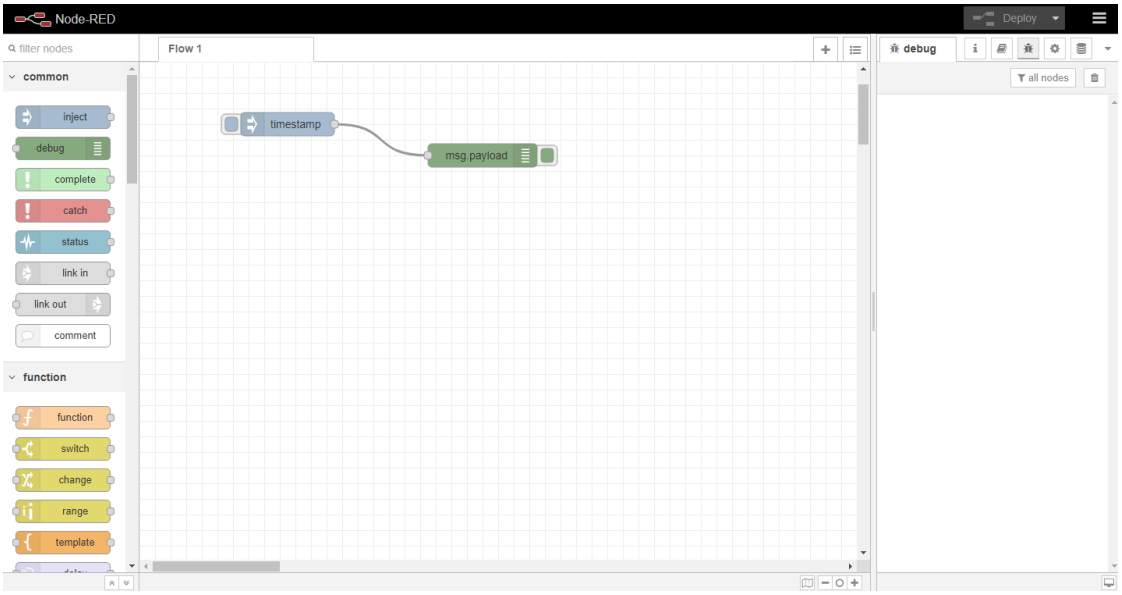
Το Node-Red προσέλκυσε μια πολύ ενεργή κοινότητα προγραμματιστών και υποστηρικτών που συμβάλλουν στην δημιουργία νέων μπλοκ κώδικα (κόμβοι). Ένας συνεχόμενα αυξανόμενος κατάλογος κόμβων ανοιχτού κώδικα επιτρέπει τη δημιουργία ακόμα και προηγμένων εφαρμογών πολύ γρήγορα, παρέχοντας ταυτόχρονα ένα οπτικό και ενθαρρυντικό μαθησιακό περιβάλλον.

Ένα ενδιαφέρον χαρακτηριστικό του Node-RED είναι ότι δίνει την δυνατότητα στους χρήστες να διαμοιράζονται εύκολα τα προγράμματα και τις ροές που δημιούργησαν. Αυτή η διαδικασία γίνεται χρησιμοποιώντας τις λειτουργίες import/export, από τις επιλογές του IDE περιβάλλοντος.

Όλες αυτές οι ιδιότητες καθιστούν το Node-Red ένα αποτελεσματικό εργαλείο για τη δημιουργία εφαρμογών που βασίζονται σε IoT. Αυτές οι εφαρμογές θα μπορούσαν να περιλαμβάνουν διάφορα σενάρια απόκτησης δεδομένων, μεταφοράς, παρακολούθησης και αναλυτικών στοιχείων.

Αν και το Node-Red φαίνεται να είναι ένα ιδανικό εργαλείο για την δημιουργία εφαρμογών από αρχάριους προγραμματιστές. Ωστόσο όπως και πολλά άλλα παρόμοια εργαλεία αν και φαίνονται ότι είναι εργαλεία για αρχαρίους, αρκετά γρήγορα απαιτούν από τους χρήστες να αναπτύξουν κάποιες περισσότερες δεξιότητες στον προγραμματισμό. Έτσι στην περίπτωση που κάποιος θέλει να δημιουργήσες μερικές πιο σημαντικές εφαρμογές θα πρέπει να μάθει την γλώσσα JavaScript για να συντάξει τα δικά του μπλοκ κώδικα. [25]

Περιήγηση στον editor του Node-Red:

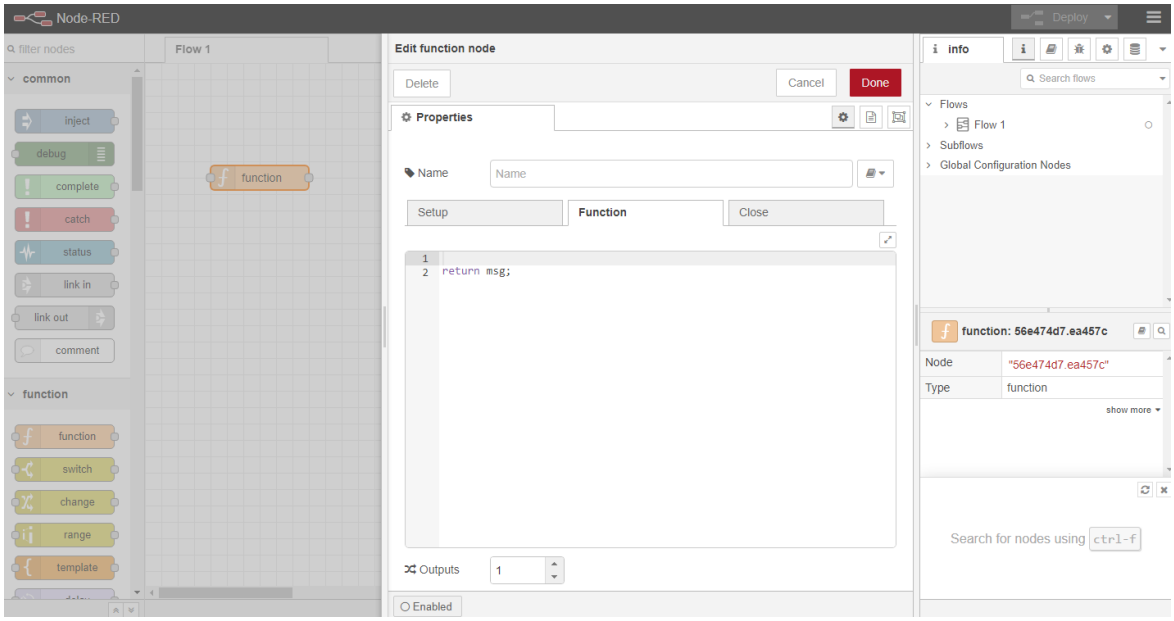


Εικόνα 22 Node-Red Editor

Για να μεταβούμε στο προγραμματιστικό περιβάλλον του Node-Red χρησιμοποιούμε ένα πρόγραμμα περιήγησης. Αριστερά του παραθύρου υπάρχει μια παλέτα με όλα τα έτοιμα μπλοκ κώδικα. Όπως αναφέρθηκε και πιο πριν υπάρχει η δυνατότητα αναβάθμισης στην συγκεκριμένη παλέτα με πρόσθετες λειτουργίες (nodes). Αρκεί να περαστεί η κατάλληλη βιβλιοθήκη και η παλέτα με τα μπλοκ θα αναβαθμιστεί με νέες κατηγορίες ή συμπληρώνοντας τις ήδη υπάρχουσες.

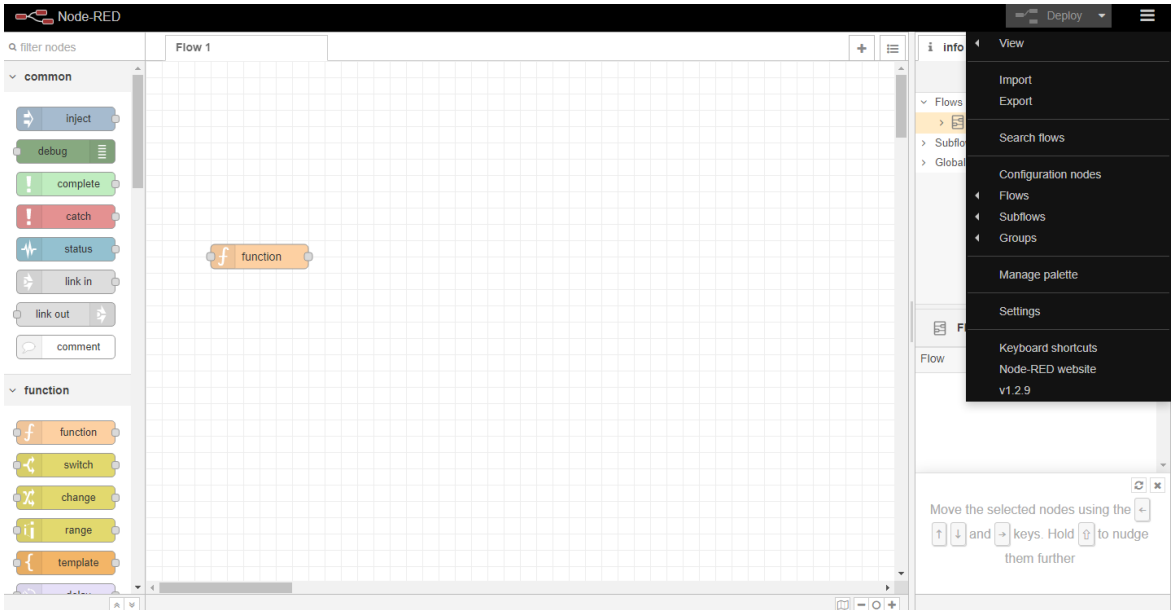
Για να εκτελέσουμε μια εργασία με το Node-Red απλώς τοποθετούμε τα μπλοκ κώδικα που μας ενδιαφέρουν και τα συνδέουμε μεταξύ τους με γραμμές. Η συνδέσεις των κόμβων ονομάζονται ροές (flows).

Σημείωση: Ο όρος "ροή" χρησιμοποιείται τόσο για την περιγραφή ενός μεμονωμένου συνόλου συνδεδεμένων κόμβων, όσο και στην αναφορά της καρτέλα (tab) που περιέχει πολλαπλές ροές (σύνολα συνδεδεμένων κόμβων). [26]



Εικόνα 23 Node-Red Function

Εκτός από τις έτοιμες λειτουργίες το Node-Red φροντίζει οι χρήστες να μπορούν να δημιουργούν τα δικά τους μπλοκ κώδικα. Αυτή η διαδικασία γίνεται χρησιμοποιώντας το μπλοκ function. Τοποθετώντας το μπλοκ πάνω στο ταμπλό και ανοίγοντάς το μας δίνετε η δυνατότητα να γράψουμε τον δικό μας κώδικα. Ο προγραμματισμός θα πρέπει να γίνει στην γλώσσα JavaScript. Αργότερα το συγκεκριμένο script μπορούμε να το ενσωματώσουμε με τις υπόλοιπες ροές του προγράμματός μας.



Εικόνα 24 Node-Red import/export

Τέλος όπως ήδη έχει αναφερθεί με το Node-Red δίνεται η δυνατότητα να διαμοιραζόμαστε τα προγράμματα που έχουμε δημιουργήσει. Αυτό γίνεται με τις επιλογές import/export από το μενού. Μπορούμε να εισάγουμε/εξάγουμε αρχεία μορφής JSON ή να αντιγράψουμε/επικολλήσουμε την αντίστοιχη JSON εντολή.

Για την εγκατάσταση του Node-Red εκτελούμε στο παράθυρο εντολών του raspberry pi τις εντολές:

sudo apt update
sudo apt upgrade
sudo apt install build-essential git
bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)

Προκειμένου να ενεργοποιείται το Node-Red αυτόματα με την ενεργοποίηση ή επανεκκίνηση του raspberry pi γράφουμε την εντολή:

sudo systemctl enable nodered.service
sudo systemctl start nodered.service

5.5 InfuxDB

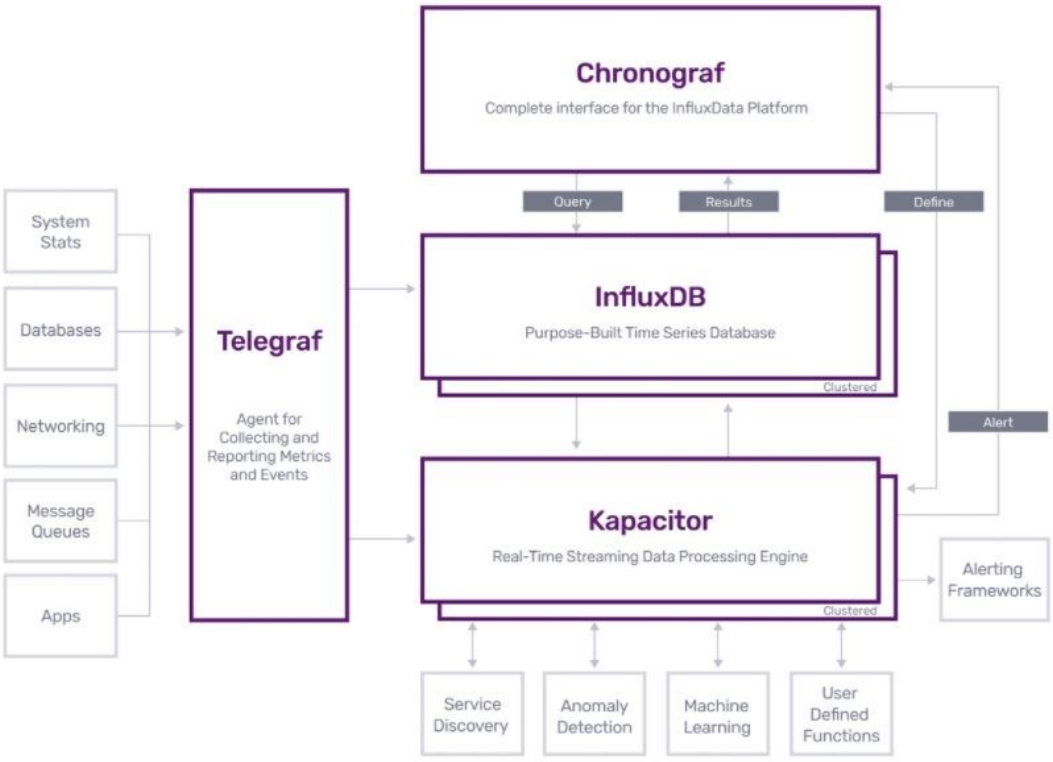
Το InfluxDB είναι μια βάση δεδομένων χρονοσειρών ανοιχτού κώδικα (TSDB) που αναπτύχθηκε από την InfluxData το 2013. Αναπτύσσεται χρησιμοποιώντας την γλώσσα προγραμματισμού Go και βελτιστοποιείται για γρήγορη αποθήκευση και ανάκτηση δεδομένων χρονοσειρών. Ανήκει στην κατηγορία NoSQL βάσεων δεδομένων και χρησιμοποιεί μια SQL-like query γλώσσα εκφραστικών ερωτημάτων, με αποτέλεσμα να προσαρμόζεται εύκολα στα ερωτήματα συγκεντρωτικών πληροφοριών.

Το InfluxDB έχει περισσότερες από 70000 ενεργές εγκαταστάσεις και προτιμάται για παρακολούθηση DevOps (παρακολούθηση υποδομής, παρακολούθηση εφαρμογών, παρακολούθηση cloud), παρακολούθηση IoT και ανάλυση δεδομένων σε πραγματικό χρόνο.

Υπάρχουν διάφοροι τρόποι εισαγωγής δεδομένων στο InuxDB, όπως το CLI, οι clients βιβλιοθήκες και τα plugins, καθώς και το ενσωματωμένο HTTP API. Θα πρέπει να σημειωθεί ότι τα δεδομένα χρονοσειρών συνήθως εισάγονται στη βάση δεδομένων σε

πραγματικό χρόνο (π.χ. δεδομένα αισθητήρα, δεδομένα αρχείου καταγραφής), οπότε το InuxDB έχει σχεδιαστεί για να χειρίζεται καλύτερα αυτά τα σενάρια πραγματικής ζωής. Για την αποθήκευση πληροφοριών χρησιμοποιεί τα πρωτόκολλα TCP, HTTP και UDP. Ένα επιπλέον πλεονέκτημα του InfluxDB είναι η ικανότητά του να συγκεντρώνει τιμές χωρίς χειροκίνητη παρέμβαση και να είναι συμβατό με πολλά εργαλεία front-end, δηλαδή εργαλεία που παρέχουν χαρακτηριστικά οπτικοποίησης δεδομένων χρονοσειρών, όπως το Grafana. [27]

Τέλος αξίζει να σημειωθεί ότι η InfluxData συνοδεύει το InfluxDB με μια συλλογή στοιχείων ανοιχτού κώδικα που συνδυάζονται για να παρέχουν μια ολοκληρωμένη πλατφόρμα αποθήκευσης, οπτικοποίησης και παρακολούθηση δεδομένων χρονοσειρών. Οπότε δίνει την δυνατότητα στον χρήστη να μην έχει ανάγκη λογισμικά τρίτων εταιριών. Αυτό το ολοκληρωμένο πακέτο λογισμικών ονομάζεται TICK Stack και αποτελείται από τα Telegraf, InfluxDB, Chronograf και Kapacitor. [28]



Εικόνα 25 TICK Stack

Telegraf: Το Telegraf είναι ένας plugin-driven server που βασίζεται σε plugins για τη συλλογή και αναφορά μετρήσεων. Τα Telegraf plugins προέρχονται από μια ποικιλία μετρήσεων απευθείας από τα συστήματα στα οποία λειτουργεί, αντλώντας μετρήσεις από

τρίτα API. Διαθέτει επίσης plugins εξόδων για την αποστολή μετρήσεων σε άλλες βάσεις δεδομένων, υπηρεσιών και ουρών μηνυμάτων, συμπεριλαμβανομένων των InfluxDB, Graphite, OpenTSDB, Datadog, Librato, Kafka, MQTT, NSQ και πολλών άλλων.

Chronograf: Το Chronograf είναι μια διεπαφή χρήστη και εργαλείο οπτικοποίησης χρονοσειρών. Διευκολύνει στην παρακολούθηση και τον έλεγχο της βάσης δεδομένων. Είναι απλό στη χρήση και περιλαμβάνει πρότυπα και βιβλιοθήκες που επιτρέπουν την γρήγορη δημιουργία dashboards με οπτικοποιήσεις σε πραγματικό χρόνο των δεδομένων και την εύκολη δημιουργία κανόνων ειδοποιήσεων και αυτοματισμού.

Kapacitor: Το Kapacitor είναι μια μηχανή επεξεργασίας native Data. Μπορεί να επεξεργαστεί δεδομένα ροής και δέσμης δεδομένων από το InfluxDB. Το Kapacitor επιτρέπει στον χρήστη να δημιουργήσει τις δικές τους συναρτήσεις για να επεξεργαστεί ειδοποιήσεις με δυναμικά όρια, να αντιστοιχίσει μετρήσεις για μοτίβα, να υπολογίσει στατιστικές ανωμαλίες και να εκτελέσει συγκεκριμένες ενέργειες βάσει αυτών των ειδοποιήσεων, όπως η δυναμική εξισορρόπηση φορτίου. Το Kapacitor ενσωματώνεται στα HipChat, OpsGenie, Alerta, Sensu, PagerDuty, Slack και άλλα.

Στο Raspberry pi 4 που χρησιμοποιήθηκε ως διακομιστής, εγκαταστάθηκε μόνο το InfluxDB επειδή αξιοποιήθηκαν λογισμικά τρίτων για τις υπόλοιπες διεργασίες όπως το Node-Red και το Grafana.

Για την εγκατάσταση του InfluxDB εκτελούμε στο παράθυρο εντολών του raspberry pi τις εντολές:

wget -qO- https://repos.influxdata.com/influxdb.key | sudo apt-key add -
echo "deb https://repos.influxdata.com/debian buster stable" | sudo tee /etc/apt/sources.list.d/influxdb.list
sudo apt update
sudo apt install -y influxdb

Προκειμένου να ενεργοποιείται το InfluxDB αυτόματα με την ενεργοποίηση ή επανεκκίνηση του raspberry pi γράφουμε την εντολή:

sudo systemctl unmask influxdb.service
sudo systemctl enable influxdb.service

5.6 Grafana

Η Grafana είναι ένα λογισμικό ανοιχτού κώδικα και πολύ δημοφιλές για οπτικοποίηση και ανάλυση στοιβών. Χρησιμοποιείται συχνά σε συνδιασμό με βάσεις δεδομένων χρονοσειρών (TSDB) όπως InfluxDB, Prometheus, Graphite και άλλες πηγές δεδομένων. Σε ένα πίνακα ελέγχου Grafana (dashboard) μπορούμε να οπτικοποιήσουμε τα αποτελέσματα από πολλές πηγές δεδομένων ταυτόχρονα και είναι δυνατόν να οριστούν κανόνες ειδοποίησης για τις βασικές μετρήσεις, έτσι ώστε να ειδοποιούμαστε αυτόματα αν κάτι δεν είναι σωστό [29] [30].

Κάποια από τα χαρακτηριστικά που κάνουν την Grafana δημοφιλής είναι τα εξής [31]:
Οπτικοποίηση: Τα Panel plugins προσφέρουν πολλούς διαφορετικούς τρόπους για να γίνει απεικόνιση μετρήσεων και αρχείων καταγραφής. Γραφήματα, ιστογράμματα και πολλά όλα βοηθούν στο να μελετιούνται καλύτερα τα δεδομένα.

Ειδοποιήσεις: Ορισμός κανόνων ειδοποίησης για τις πιο σημαντικές μετρήσεις. Η Grafana αξιολογεί συνεχώς τα δεδομένα και στέλνει ειδοποιήσεις σε συστήματα όπως Slack, PagerDuty, VictorOps, OpsGenie.

Δυναμικοί πίνακες ελέγχου: Δημιουργεί δυναμικούς και επαναχρησιμοποιήσιμους πίνακες ελέγχου με μεταβλητές προτύπου που εμφανίζονται σε αναπτυσσόμενο μενού στο πάνω μέρος του πίνακα ελέγχου.

Εξερεύνηση αρχείων καταγραφής: Εύκολη μετάβαση μετρήσεων σε αρχεία καταγραφής με διατηρημένα φίλτρα ετικετών και γρήγορη αναζήτηση σε αυτά.

Μικτές πηγές δεδομένων: Είναι δυνατή η ανάμιξη διαφορετικών πηγών δεδομένων στο ίδιο γράφημα. Μπορεί να καθορίστει μια πηγή δεδομένων ανά ερώτημα. Αυτό λειτουργεί ακόμη και για προσαρμοσμένες πηγές δεδομένων.

Εξερεύνηση μετρήσεων: Εξερεύνηση δεδομένων μέσω ad-hoc ερωτημάτων και δυναμικής ανάλυσης. Διαχωρίστε την προβολή και συγκρίνετε διαφορετικά χρονικά διαστήματα, ερωτήματα και πηγές δεδομένων δίπλα-δίπλα.

Ανοικτού Κώδικα: Είναι ένα λογισμικό ανοιχτού κώδικα. Μπορεί να χρησιμοποιηθεί το Grafana Cloud ή να το εγκατασταθεί εύκολα σε οποιαδήποτε πλατφόρμα.

Για την εγκατάσταση του Grafana εκτελούμε στο παράθυρο εντολών του raspberry pi τις εντολές:

wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -
echo "deb https://packages.grafana.com/oss/deb stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list
sudo apt update
sudo apt install -y grafana

Προκειμένου να ενεργοποιείται η Grafana αυτόματα με την ενεργοποίηση ή επανεκκίνηση του raspberry pi γράφουμε την εντολή:

sudo systemctl enable grafana-server.service
sudo systemctl start grafana-server

Αναλυτικές πληροφορίες εγκατάστασης του εξυπηρετητή παρατίθενται στο Παράρτημα Β.

6 παρουσίαση Εφαρμογών Διεπαφής Χρήστη

Στο συγκεκριμένο κεφάλαιο θα παρουσιαστεί ο τρόπος με τον οποίο ο χρήστης μπορεί να ελέγξει και να αναλύσει τις μετρήσεις του συστήματος μέσω δύο δεπαφών χρήστη. Για την δημιουργία των διεπαφών θα χρησιμοποιηθούν δύο διαφορετικά λογισμικά το Node-Red και το Grafana. Ο λόγος για τον οποίο επιλέχτηκαν τα συγκεκριμένα λογισμικά είναι η απαίτηση στον εντοπισμό εργαλείων με τα οποία ο οποιοσδήποτε ανεξαρτήτως του μεγέθους των προγραμματιστικών του γνώσεων να μπορεί να δημιουργήσει δύο αξιόλογα dashboard.

6.1 Παρουσίαση Εφαρμογής με Node-Red

6.1.1 Dashboard του Node-Red

Με το Node-red δημιουργήθηκε ένα dashboard που σκοπό έχει τον άμεσο χειρισμό του συστήματος θέρμανσης. Για να συνδεθούμε στο dashboard πληκτρολογούμε την διεύθυνση «http: // <ip address>: 1880/ui/», όπου στην θέση του <ip address> βάζουμε την διεύθυνση IP του Raspberry Pi.

Το πρώτο πράγμα που εμφανίζεται στην οθόνη του χρήστη είναι η «ΚΕΝΤΡΙΚΗ ΣΕΛΙΔΑ» όπου ο χρήστης μπορεί να διαβάσει διάφορες πληροφορίες. Όπως φαίνεται στην εικόνα 26 οι πληροφορίες αυτές είναι:

- Η θερμοκρασία/υγρασία του εσωτερικού χώρου όπως την στέλνει το Node 1.
- Η θερμοκρασία/υγρασία του εξωτερικού χώρου και την ηλιακή ακτινοβολία όπως την στέλνει το Node 5.
- Η θερμοκρασία του θερμαντικού σώματος όπως στέλνετε από το Node 2.

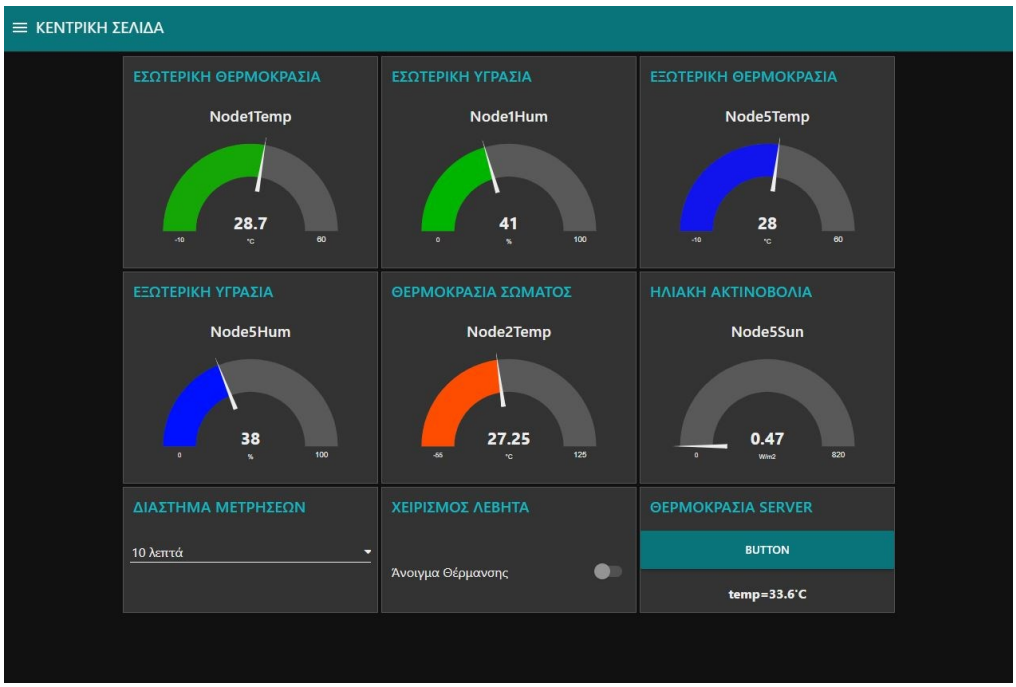
Οι συγκεκριμένες οθόνες «gauge» δείχνουν την τελευταία τιμή που στάλθηκε από τις πλακέτες διαμέσου του πρωτοκόλλου MQTT.

Εκτός από τις οπτικές οθόνες παρουσιάζονται και 3 οθόνες επιλογών χειρισμού. Σε αυτές τις οθόνες δίνουμε την δυνατότητα στον χρήστη να διαχειρίζεται το σύστημα, διαμέσου της εφαρμογής. Στην πρώτη οθόνη με το όνομα «Διάστημα Μετρήσεων» δίνεται η δυνατότητα στον χρήστη να επιλέξει το χρονικό διάστημα που αντιστοιχεί ανάμεσα στην αποστολή των

μετρήσεων των πλακετών, δηλαδή ο χρήστης μπορεί να επιλέξει κάθε πότε να γίνεται η καταμέτρηση των αισθητήρων και η αποστολή των δεδομένων στον MQTT Broker.

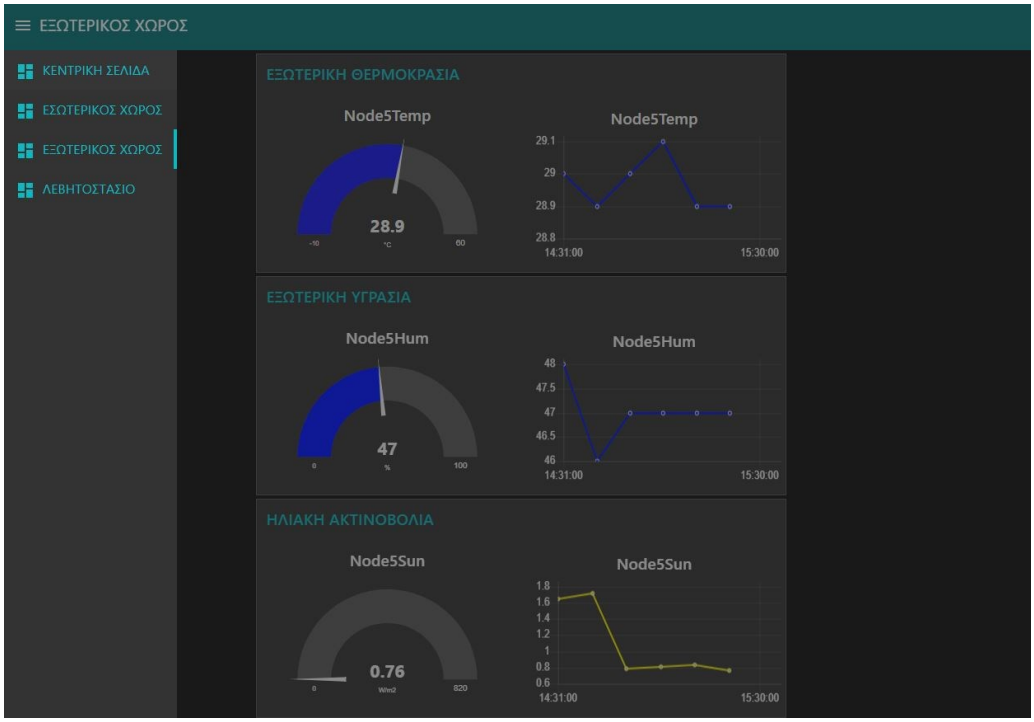
Στην δεύτερη οθόνη με το όνομα «Χειρισμός Λέβητα», δίνεται η δυνατότητα στον χρήστη να χειρίζεται το relay που είναι συνδεδεμένο με την πλακέτα Node 6 του υποσυστήματος του λεβητοστασίου. Έτσι ο χειριστής μπορεί να ανοίγει και να κλείνει την θέρμανση του κτηρίου.

Τέλος στην οθόνη με το όνομα «Θερμοκρασία Server», ο χρήστης μπορεί πατώντας την επιλογή «Button», να βλέπει την τρέχουσα θερμοκρασία της CPU του raspberry pi 4, δηλαδή του Server.



Εικόνα 26 Node-Red «ΚΕΝΤΡΙΚΗ ΣΕΛΙΔΑ»

Εκτός από την «ΚΕΝΤΡΙΚΗ ΣΕΛΙΔΑ» δημιουργήθηκαν οι καρτέλες «ΕΣΩΤΕΡΙΚΟΣ ΧΩΡΟΣ», «ΕΞΩΤΕΡΙΚΟΣ ΧΩΡΟΣ» και «ΛΕΒΗΤΟΣΤΑΣΙΟ». Για να μεταβούμε στις συγκεκριμένες καρτέλες επιλέγουμε τις γραμμές στην πάνω αριστερή γωνία ώστε να ανοίξει ο αναδυόμενος κατάλογος. Η καθεμιά συγκεντρώνει πληροφορίες για τους αισθητήρες του κάθε υποσυστήματος, συγκεκριμένα για τον κάθε αισθητήρα έχουμε μια οθόνη «gauge» που φανερώνει την τρέχουσα τιμή του αισθητήρα και μια οθόνη «chart» που δείχνει το σύνολο τιμών για ένα χρονικό διάστημα. Οι οθόνες «chart» και «gauge» που αναφέρονται στον ίδιο αισθητήρα ορίζονται σαν μια κοινή ομάδα (group).



Εικόνα 27 Node-Red Menu

Σε αυτή την περίπτωση πρέπει να τονιστεί ότι επειδή το διάγραμμα chart του Node-Red έχει εντολή να παίρνει τις τιμές από τον MQTT Broker, αν κλείσει ο Server τότε μηδενίζεται και χάνει τις τιμές του. Αυτό γίνεται επειδή ο MQTT Broker δεν συγκρατεί τις τιμές μόλις επιτελέσουν τον σκοπό τους. Αντίθετα στο dashboard του Grafana, τα αντίστοιχα «chart» διαγράμματα, ακόμα και αν κλείσει ο Server, επαναφέρουν τις τιμές τους, μιας και ενημερώνονται από το InfluxDB το οποίο αποθηκεύει όλες τις τιμές που του στέλνονται με χρονική σήμανση.

Το dashboard του Node-Red έχει την ικανότητα να προσαρμόζεται ανάλογα με την οθόνη της συσκευής που χρησιμοποιούμε. Γι αυτό πρέπει να δίνεται προσοχή στο πως οργανώνονται οι ομάδες με τους διάφορους πίνακες. Αυτό που ουσιαστικά κάνει το Node-Red είναι να παίρνεις τις ομάδες και να τις κατατάσσει από πάνω αριστερά προς τα δεξιά με βάσει τα όρια της καρτέλας που έχουμε ορίσει.

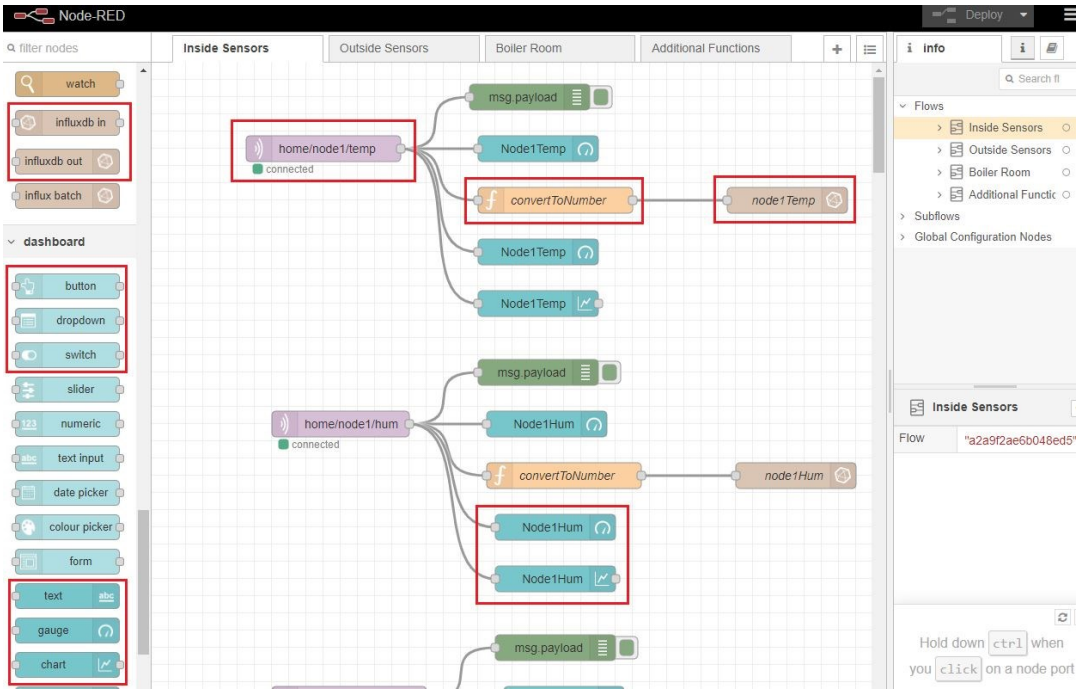
6.1.2 Πρόγραμμα στο Node-Red

Όπως ήδη έχει αναφερθεί στο κεφάλαιο 5 το Node-Red χρησιμοποιεί οπτικό προγραμματισμό και ο προγραμματιστής καλείται να συνδέσει μεταξύ τους προκαθορισμένα μπλοκ κώδικα προκειμένου να εκτελεστεί μια διεργασία. Στην

συγκεκριμένη ενότητα θα παρουσιαστεί το πρόγραμμα που έχει δημιουργηθεί για το συγκεκριμένο σύστημα με την χρήση του Node-Red.

Για να μεταβούμε στο προγραμματιστικό περιβάλλον (IDE) του Node-Red σε ένα πρόγραμμα περιήγησης γράφουμε την διεύθυνση «http: // <ip address>: 1880», όπου στην θέση του <ip address> βάζουμε την διεύθυνση IP του Raspberry Pi.

Το πρώτο πράγμα που εμφανίζεται στην οθόνη είναι ο κώδικας που βρίσκεται χωρισμένος σε 4 «tabs». Έχουμε το «Inside Sensors» το οποίο περιέχει ότι υλοποιήσεις αφορούν το υποσύστημα του εσωτερικού χώρου, το «Outside Sensors» το οποίο περιέχει τις υλοποιήσεις του υποσυστήματος εξωτερικού χώρου, το «Boiler Room» το οποίο περιέχει τις υλοποιήσεις του υποσυστήματος λεβητοστασίου και τέλος το tab «Additional Functions» στο οποίο αναφερόμαστε στις γενικές λειτουργικότητες της εφαρμογής.

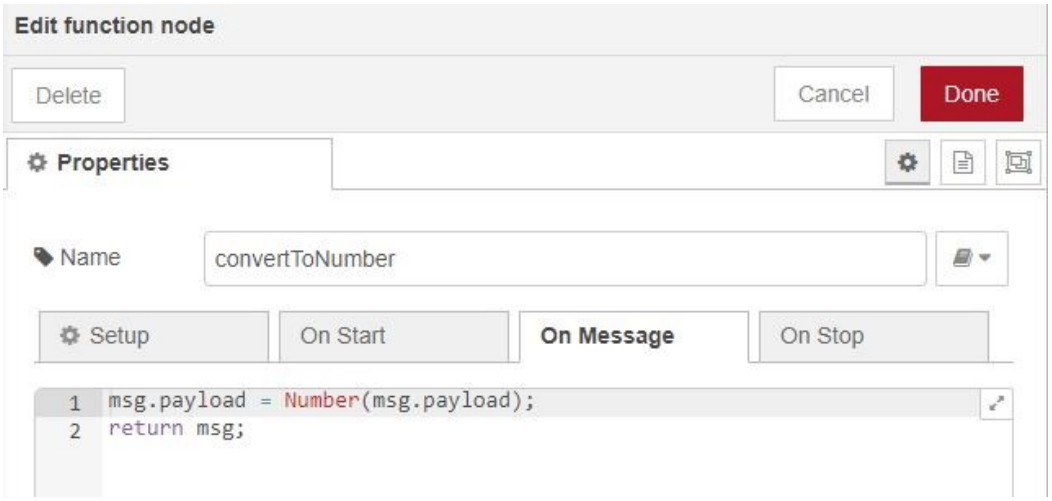


Εικόνα 28 Node-Red IDE

Για όλους τους αισθητήρες χειριζόμαστε τα δεδομένα με τον ίδιο τρόπο. Αρχικά χρησιμοποιούμε κόμβους «mqtt in» οι οποίοι έχουν τον ρόλο του MQTT client και συλλέγουν τις μετρήσεις των αισθητήρων κάνοντας subscribe στο αντίστοιχο topic του MQTT Broker. Στην συνέχεια τα δεδομένα οπτικοποιούνται στους πίνακες «gauge» και «chart» του dashboard χρησιμοποιώντας τους αντίστοιχους κόμβους από την παλέτα. Ταυτόχρονα διαμέσου ενός κόμβου «function» και γράφοντας τον απαραίτητο κώδικα σε java script, μετατρέπουμε τα εισερχόμενα δεδομένα από αλφαριθμητικά σε αριθμητικά και

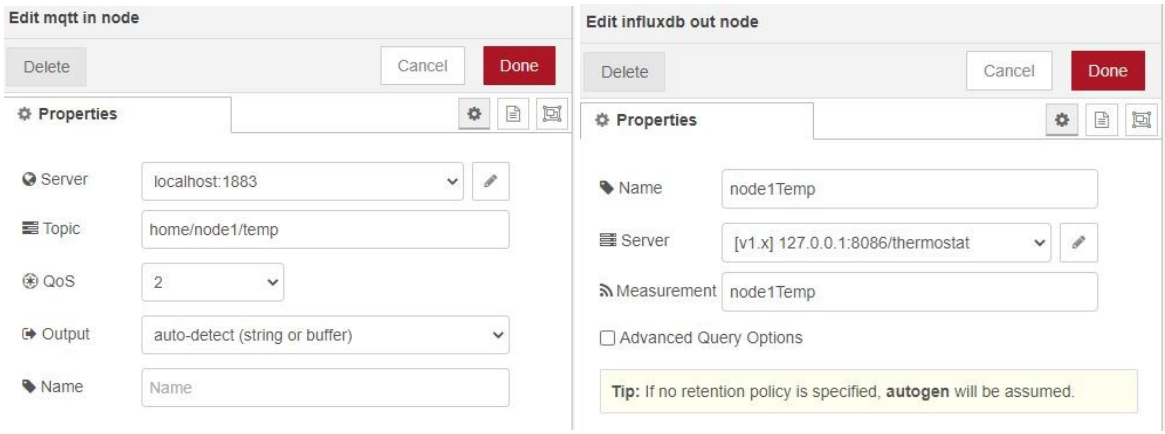
κατόπιν τα στέλνουμε στην βάση δεδομένων influxDB διαμέσου του κόμβου «influxDB out».

Γνωρίζουμε από το κεφάλαιο 3 ότι τα δεδομένα διαμέσου των μηνυμάτων του πρωτοκόλλου MQTT μεταφέρονται σε μορφή αλφαριθμητικού. Ωστόσο το Grafana δεν μπορεί να οπτικοποιήσει αλφαριθμητικά δεδομένα στα δικά της «Chart» διαγράμματα. Αυτό έχει ως αποτέλεσμα να πρέπει να γίνει μετατροπή σε αριθμητικά προτού σταλούν στην βάση δεδομένων influxDB και αργότερα αντληθούν από το Grafana. Για να επιτευχθεί αυτός ο σκοπός στον κόμβο συνάρτηση γράφουμε τον κώδικα που φαίνεται στην εικόνα 29.



Εικόνα 29 Node-Red Function Node

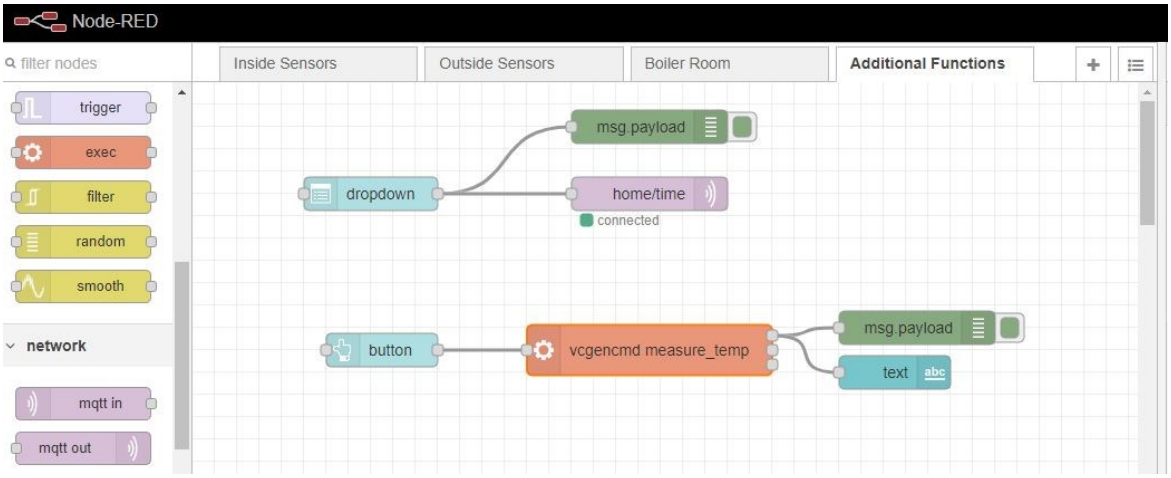
Τόσο το «mqtt in» όσο και το «influxDB out» δημιουργούν αντικείμενα Client προκειμένου να κάνουν δυνατή την επικοινωνία του Node-Red με τους αντίστοιχους διακομιστές. Όπως είναι φυσικό για να συνδεθούν χρειάζεται στις ρυθμίσεις να τους δοθούν τα απαραίτητα στοιχεία, η διεύθυνση IP των Server και στοιχεία πιστοποίησης εισόδου, αν χρειάζονται. Για την συγκεκριμένη εργασία ως διεύθυνση IP χρησιμοποιήθηκε η ονομασία localhost και την αντίστοιχη πόρτα του κάθε διακομιστή, από την στιγμή που όλα τα λογισμικά είναι εγκατεστημένα στην ίδια συσκευή. Επιπλέον για την διασύνδεση με το IndluxDB θα πρέπει να οριστεί το όνομα της βάσης δεδομένων, «thermostat» καθώς και το όνομα που θα έχει το αντίστοιχο measurement το οποίο θα δημιουργηθεί κατά την μεταφορά της τιμής.



Εικόνα 30 Node-Red Client MQTT – Client InfluxDB

Τέλος θα αναφερθούμε στο τελευταίο tab με το όνομα «Additional Functions». Στο συγκεκριμένο tab υλοποιούνται δύο ροές κόμβων. Στην πρώτη δημιουργούμε ένα παράθυρο με το οποίο ο χρήστης επιλέγει μια αριθμητική τιμή από το 1 ως το 15 και αργότερα το node-red την δημοσιεύει στον MQTT Broker. Η συγκεκριμένη τιμή αφορά το διάστημα που αντιστοιχεί ανάμεσα στις αποστολές των δεδομένων από τις πλακέτες ESP32.

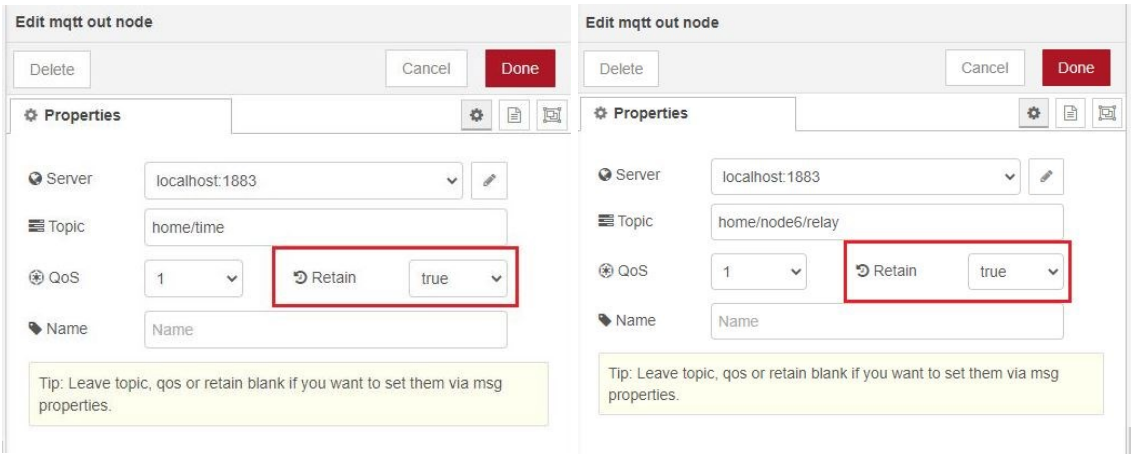
Στην δεύτερη ροή χρησιμοποιείται ο κόμβος «exec» οποίος έχει την ικανότητα να εκτελεί εντολές συστήματος. Στην συγκεκριμένη ροή έχουμε οπτικοποιήσει ένα πλήκτρο στο dashboard και όποτε ο χρήστης το πατά τότε εκτελείται η εντολή «vcgencmd measure_temp» όπου μας εμφανίζει την θερμοκρασία της CPU του διακομιστή.



Εικόνα 31 Node-Red Additional Functions

Σε αυτό το σημείο, πρέπει να διευκρινιστεί μια επιλογή που κάναμε για την δημοσίευση της τιμής στο θέμα «home/time». Όπως αναφέρθηκε στο κεφάλαιο 3 στους MQTT Broker τα μηνύματα χάνονται μετά την αποστολή τους. Τόσο στο συγκεκριμένο topic όσο και στο

topic με την ονομασία «home/node6/relay» έχουμε επιλέξει την ρύθμιση διαμέσου της σημαίας «retain» το μήνυμα να είναι «διατηρούμενο μήνυμα». Αυτό έχει ως αποτέλεσμα να διατηρείται η τελευταία τιμή στα συγκεκριμένα topic. Με αυτό τον τρόπο οποιαδήποτε στιγμή και αν συνδεθεί η αντίστοιχη πλακέτα μετά από αποσύνδεση τότε θα λάβει με την μία την τιμή που αναγράφεται και θα συνεχίσει την λειτουργία της.



Εικόνα 32 Node-Red topic time - relay

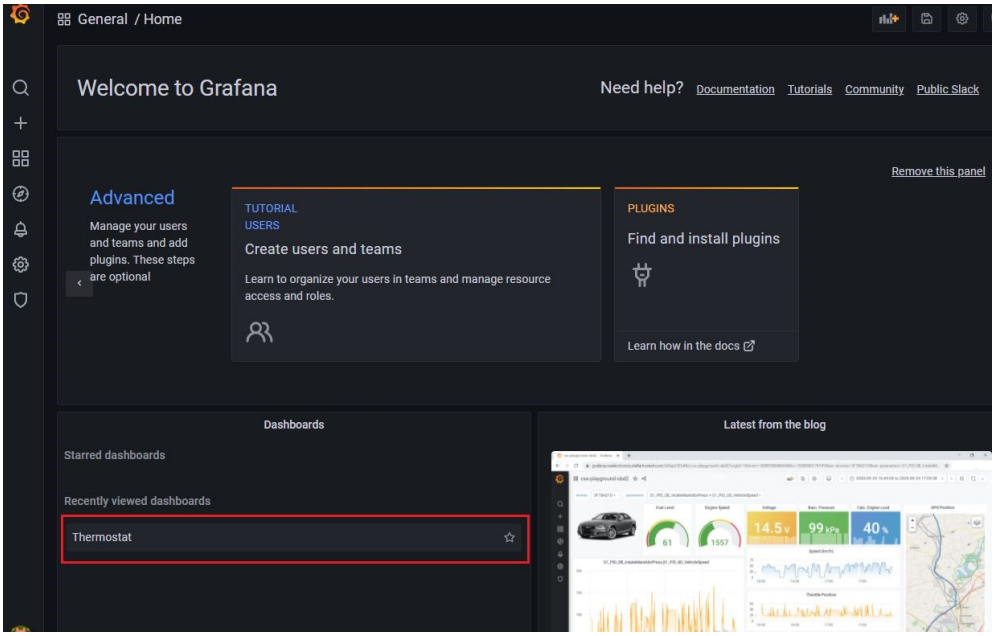
Ο κώδικας που υλοποιήθηκε στο Node-Red βρίσκεται στο «Παράρτημα Γ» σε μορφή Compact Json.

6.2 Παρουσίαση Εφαρμογής με Grafana

6.2.1 Dashboard του Grafana

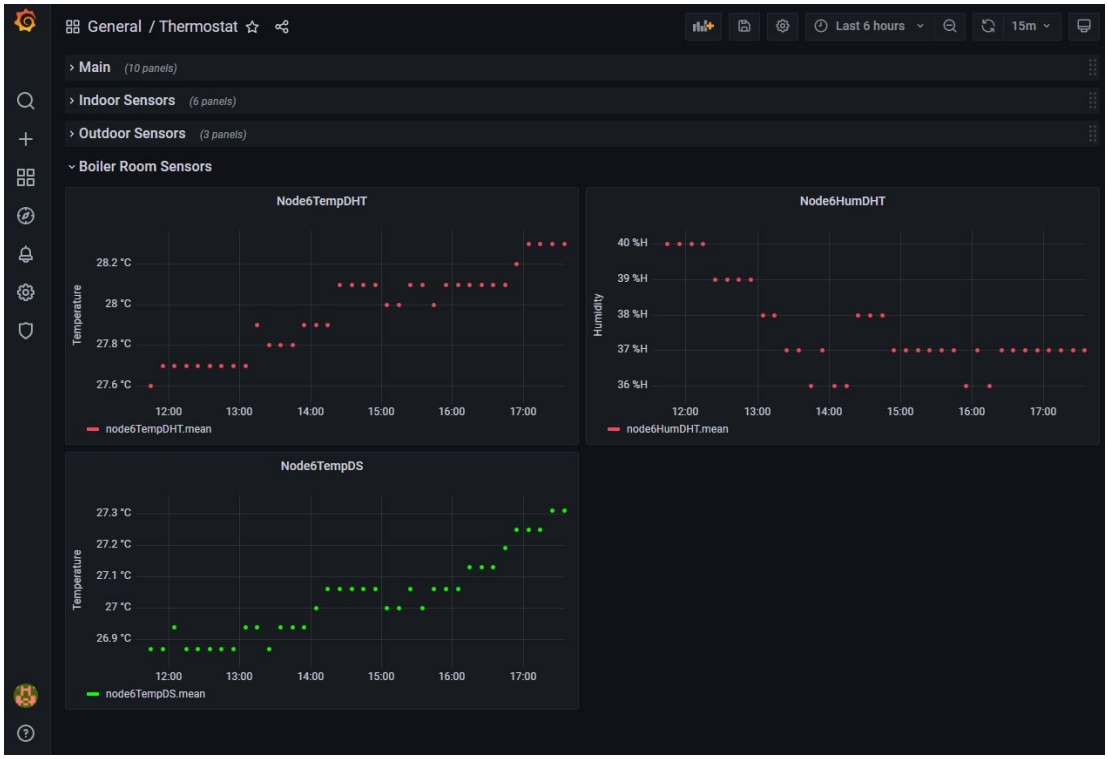
Αντίθετα με το Node-Red το dashboard που δημιουργήθηκε με το Grafana έχει ως σκοπό την οπτικοποίηση των δεδομένων ώστε να είναι δυνατή η ανάλυση και η μελέτη τους. Για να συνδεθούμε στο dashboard του Grafana πληκτρολογούμε την διεύθυνση «http: // <ip address>: 3000», όπου στην θέση του <ip address> βάζουμε την διεύθυνση IP του Raspberry Pi.

Στη συνέχεια, εμφανίζεται η αρχική σελίδα όπου είναι απαραίτητη η εισαγωγή «username» και «password» για να μπορέσει ο χρήστης να λάβει πρόσβαση στην εφαρμογή. Αφού επιβεβαιωθεί η είσοδος του χρήστη, εμφανίζεται η κεντρική σελίδα του grafana και από εκεί μπορούμε να πλοηγηθούμε στο dashboard που έχουμε δημιουργήσει. Στην συγκεκριμένη περίπτωση θα έχει το όνομα «Thermostat».



Εικόνα 33 Grafana Αρχική Σελίδα

Το dashboard του «Thermostat», αποτελείται από πίνακες «chart» και «gauge». Όπως στο node-red έτσι και στο grafana μπορούμε να τους ομαδοποιήσουμε σε ομάδες με το όνομα «row». Το dashboard χωρίζεται σε 4 ομάδες, την «Main», την «Indoor Sensors», την «Outdoor Sensors» και την «Boiler Room Sensors». Όλες εκτός της «Main» αναφέρονται στο καθένα από τα 3 υποσυστήματα αισθητήρων που έχουμε δημιουργήσει και περιέχουν ένα chart για την κάθε τιμή που στέλνεται από τον αντίστοιχο αισθητήρα. Όπως φαίνεται στην επόμενη εικόνα τα συγκεκριμένα chart είναι ρυθμισμένα να αναφέρονται στην ιδιότητα της τιμής, αν είναι δηλαδή θερμοκρασία ή υγρασία και στην πλακέτα από την οποία προέρχεται. Έχουμε ρυθμίσει οι τιμές να εμφανίζονται ως σημεία μιας και ο ρόλος των συγκεκριμένων πινάκων είναι ο εντοπισμός στην απώλεια μηνυμάτων από τις αντίστοιχες πλακέτες. Με αυτόν τον τρόπο μπορούμε να βλέπουμε εύκολα πότε και ποια πλακέτα δεν έστειλε κάποια καταμέτρηση και αν έχει κάποιο πρόβλημα σε περίπτωση που χάνονται συνεχόμενα πολλές τιμές.



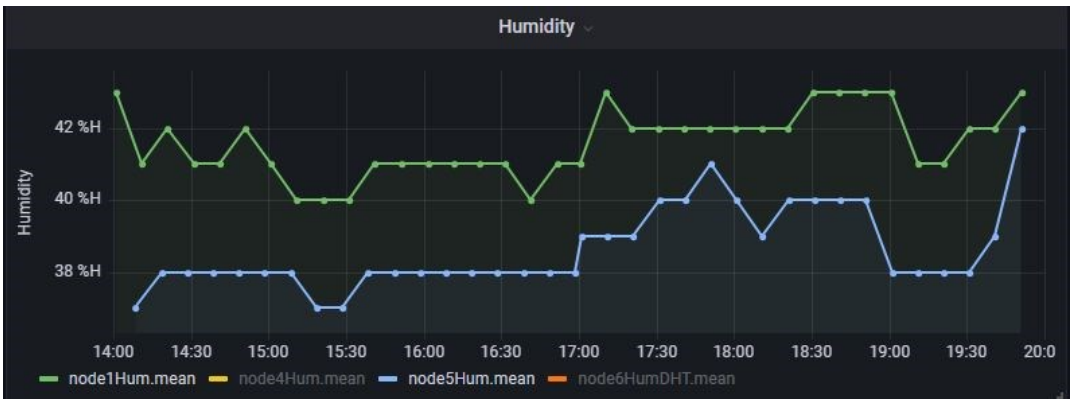
Εικόνα 34 Grafana Boiler Room Sensors

Στην «Main» έχουμε δημιουργήσει πίνακες με συνδυαστική αναπαράσταση τιμών, ώστε να καταλήγουμε σε συμπεράσματα όσο αφορά την συνολική λειτουργία του θερμοστάτη αναλύοντας τα αντίστοιχα δεδομένα.

Αρχικά υπάρχουν δύο πίνακες «chart» οι οποίοι περιέχουν τις μετρήσεις για υγρασία και θερμοκρασία αντίστοιχα, όλων τον πλακετών που έχουν τον αισθητήρα DHT11. Οι συγκεκριμένες πλακέτες καταμετρούν τις περιβαλλοντικές συνθήκες του εσωτερικού, εξωτερικού χώρου και λεβητοστασίου. Ο πρώτος πίνακας «Chart» οπτικοποιεί τις μετρήσεις που αναφέρονται στην υγρασία ανά χρονικές περιόδους και δεξιά του βρίσκονται οι τελευταίες τιμές που στάλθηκαν από τους αισθητήρες οι οποίες φανερώνονται με πίνακες «gauge». Κατά παρόμοιο τρόπο οπτικοποιούνται οι μετρήσεις που αναφέρονται στην θερμοκρασία με την μόνη διαφορά ότι οι πίνακες «gauge» βρίσκονται από κάτω.

Εικόνα 35 Grafana Dashboard

Αν και τις περισσότερες φορές δεν χρειάζεται να συγκριθούν και οι 4 τιμές ταυτόχρονα, μια ιδιότητα της εφαρμογής Grafana είναι ότι επιτρέπει στο χρήστη να επιλέξει κάθε φορά τις μεταβλητές που τον ενδιαφέρουν. Πατώντας ctrl και αριστερό κλικ στο όνομα της καταμέτρησης που μας ενδιαφέρει απαλείφουμε τις περιττές καταμετρήσεις και συγκρίνουμε όποιες επιθυμούμε.



Εικόνα 36 Grafana Επιλογή Μεταβλητών

Αμέσως μετά στην ομάδα «Main» δημιουργήσαμε ένα πίνακα «chart» που οπτικοποιεί τις τιμές θερμοκρασίας που προέρχονται από τις πλακέτες που διαθέτουν τον αισθητήρα DS18B20. Ο συγκεκριμένος πίνακας έχει ως σκοπό να φανερώνει την θερμοκρασία του νερού σε όλη την διαδρομή που ακολουθεί μέχρι να φτάσει στο θερμαντικό σώμα. Δεξιά

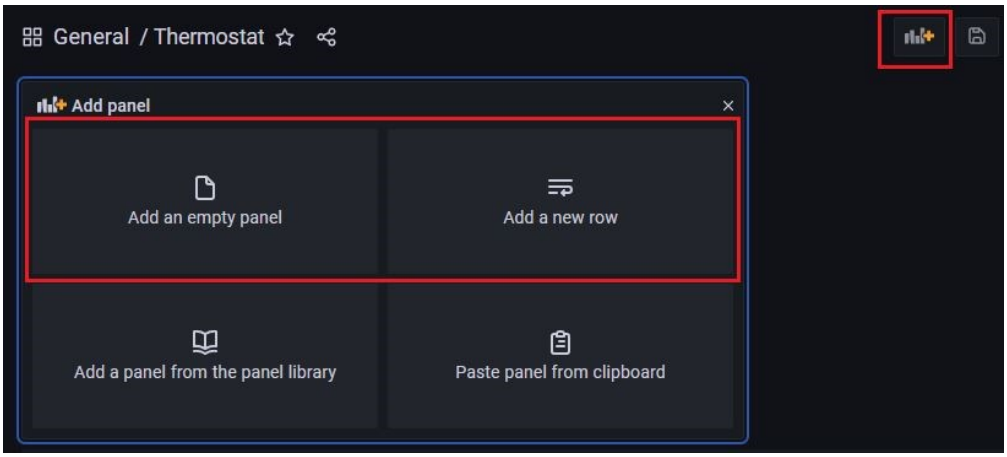
στον συγκεκριμένο πίνακα έχει δημιουργηθεί ένα «gauge char» όπου δείχνει τις τελευταίες τιμές που κατέγραψαν οι αισθητήρες.

Τέλος υπάρχει ένας πίνακας που κάνει αναφορά στην ηλιακή ακτινοβολία και οπτικοποιεί τα δεδομένα που στάλθηκαν από το Node 5 του εξωτερικού χώρου.

Εικόνα 37 Grafana DS18B20 και BH1750

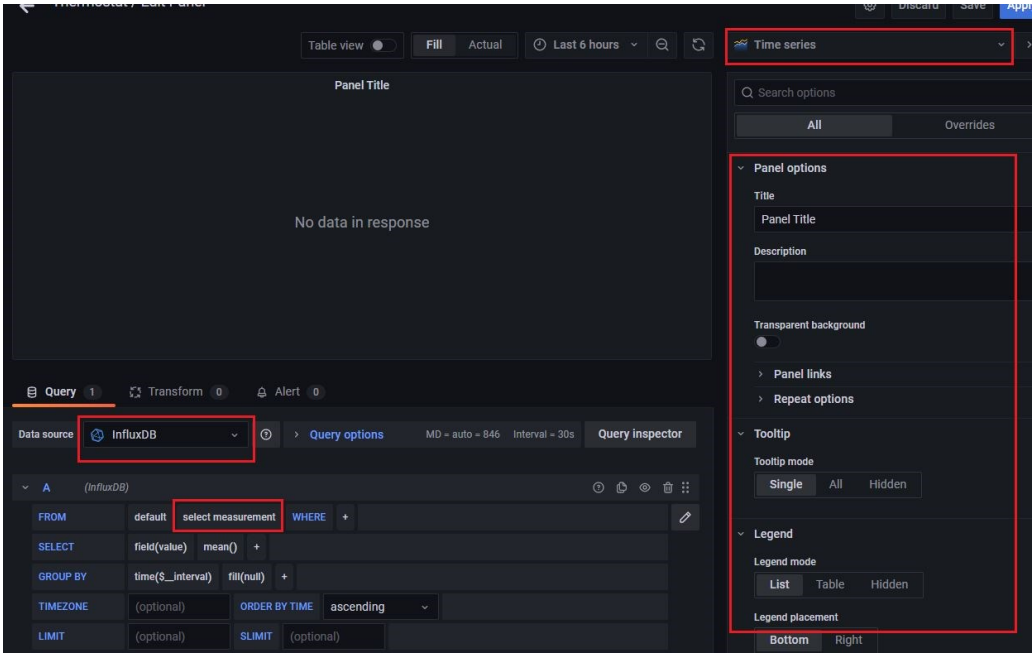
6.2.2 Περιβάλλον Ρυθμίσεων Πίνακα και Εξαγωγή Αρχείου CSV

Για την δημιουργία νέων στοιχείων στο dashboard του Grafana επιλέγουμε στην πάνω δεξιά γωνία το «add panel». Από τον πίνακα που θα εμφανιστεί μας δίνεται η δυνατότητα να επιλέξουμε την δημιουργία ενός νέου πίνακα ή μιας καινούργιας ομάδας.



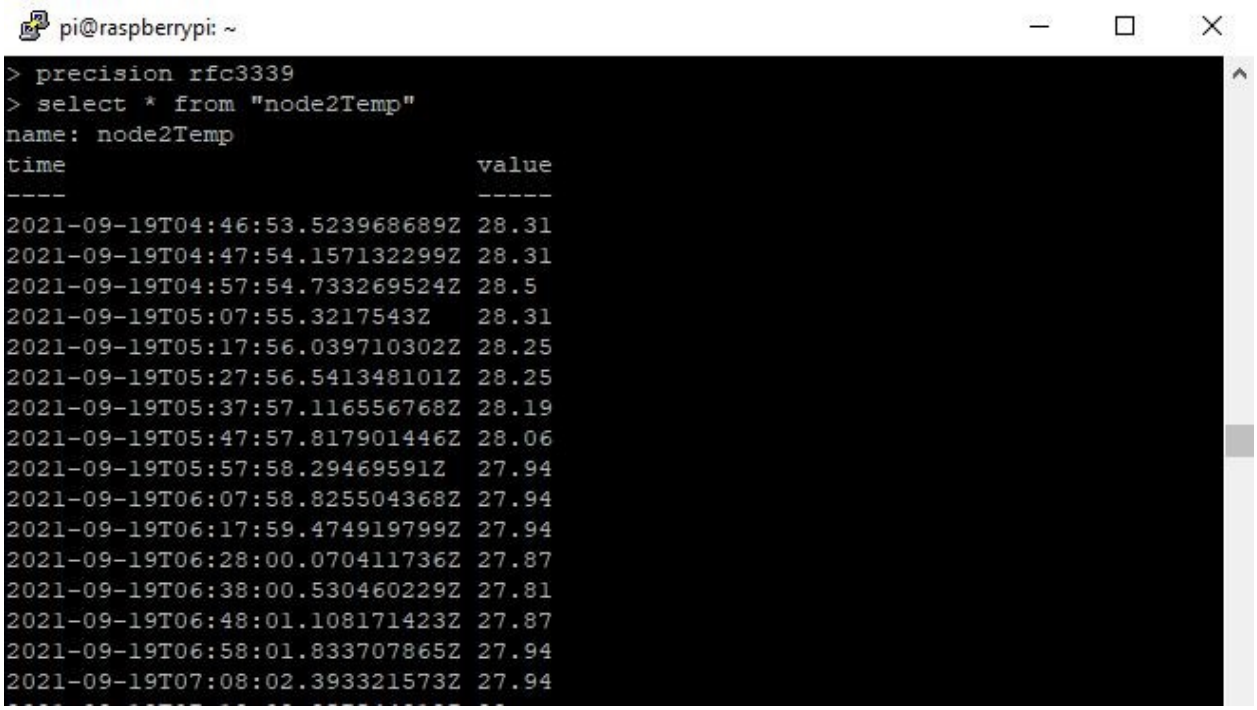
Εικόνα 38 Grafana Create New Panel

Επιλέγοντας την δημιουργία νέου πίνακα εμφανίζεται ένα παράθυρο με όλες τις απαραίτητες ρυθμίσεις προκειμένου να φτάσουμε στο επιθυμητό οπτικό αποτέλεσμα. Στην αριστερή μεριά του παραθύρου έχουμε τις επιλογές που αφορούν την βάση δεδομένων από την οποία θα αντλήσουμε τα δεδομένα. Στην πάνω αριστερή μεριά επιλέγουμε την βάση που έχουμε καταχωρημένη στο grafana και αργότερα καθορίζουμε τα ερωτήματα SQL-Lite που θέλουμε να κάνουμε. Στην πάνω δεξιά μεριά επιλέγουμε την κατηγορία πίνακα όπως «gauge», «chart» και άλλα. Στην κάτω δεξιά μεριά καθορίζουμε τις ρυθμίσεις οπτικοποίησης που θα έχει ο συγκεκριμένος πίνακας.



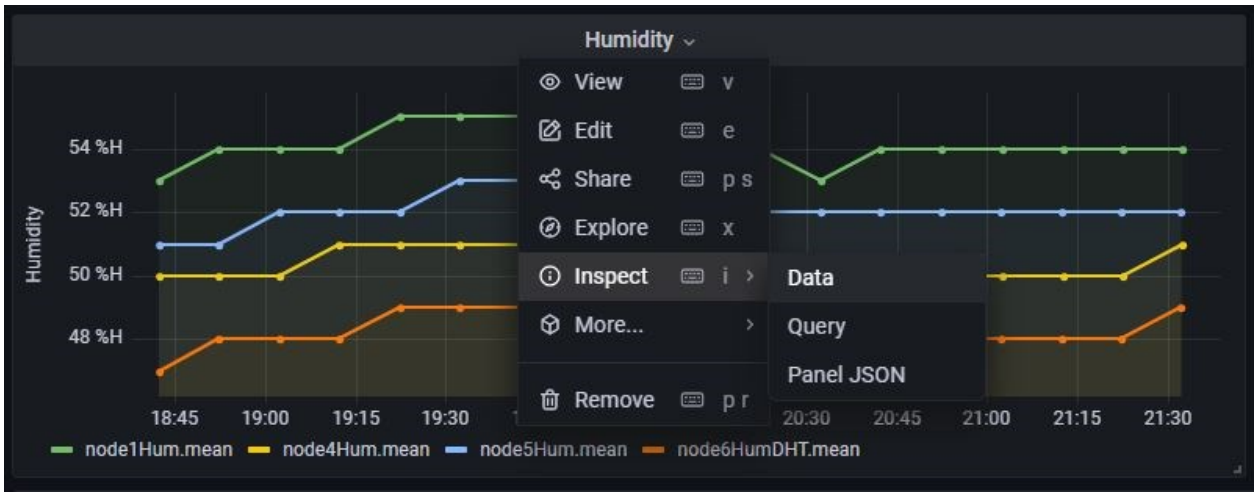
Εικόνα 39 Grafana Panel Settings

Πρέπει να τονιστεί ότι ανάλογα με το ποια βάση δεδομένων χρησιμοποιείται, τα ερωτήματα προσαρμόζονται σε αυτήν. Για παράδειγμα στην συγκεκριμένη περίπτωση χρησιμοποιείται ως πηγή η βάση δεδομένων InfluxDB. Στην συγκεκριμένη βάση χρονοσειρών δεν δίνεται η δυνατότητα να δημιουργηθούν πίνακες. Αντί για πίνακες το influxDB έχει στοιχεία που ονομάζονται «measurement». Κάθε «measurement» περιέχει μια στήλη με την χρονική σήμανση που αποθηκεύτηκε η τιμή και μια στήλη με την ίδια την τιμή. Ο χρήστης ονομάζει το «measurement» προκειμένου να γνωρίζει που αναφέρονται οι τιμές. Αυτό έχει ως αποτέλεσμα το ερώτημα που διαμορφώνουμε με την Grafana να αναφέρεται μόνο στην μοναδική στήλη του συγκεκριμένου «measurement».



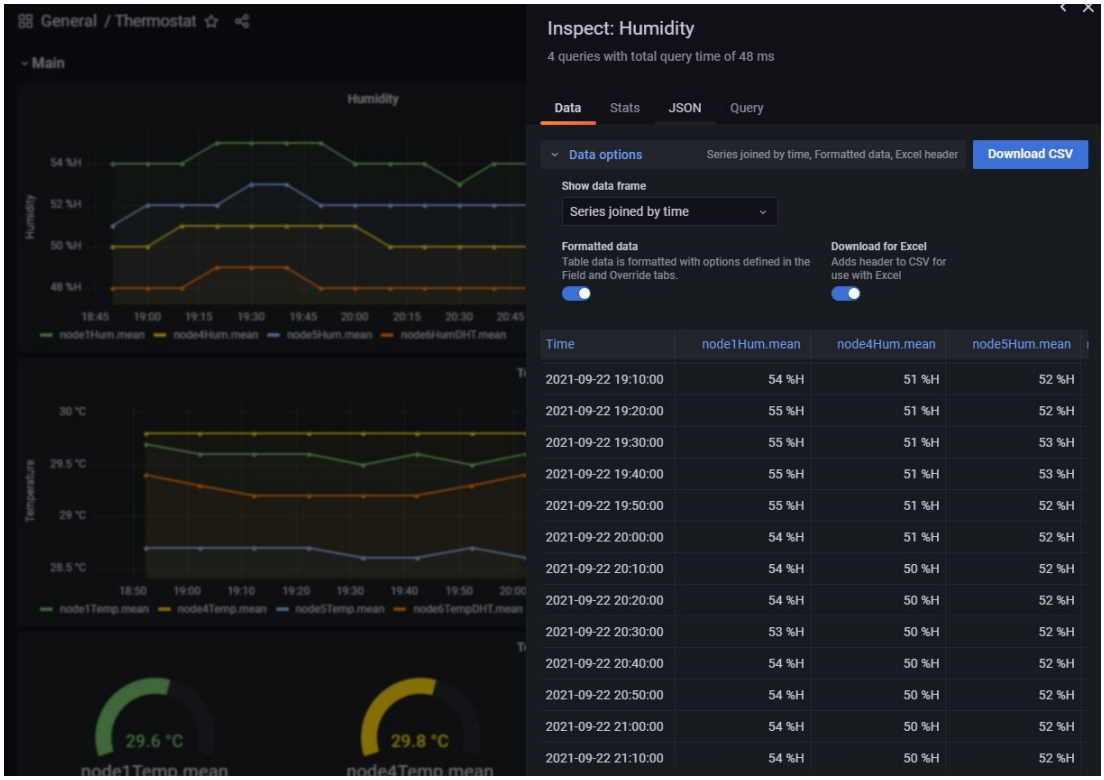
Εικόνα 40 InfluxDB

Τέλος ένα από τα βασικότερα χαρακτηριστικά του Grafana είναι η δυνατότητα εξαγωγής αρχείου CSV. Με αυτό τον τρόπο μας δίνεται η δυνατότητα να συλλέξουμε τα δεδομένα που επιθυμούμε και αφού τα εξάγουμε να προχωρήσουμε σε περεταίρω ανάλυση αυτών διαμέσου άλλων λογισμικών. Για να εξαχθούν τα δεδομένα που οπτικοποιεί κάποιος πίνακας, ο χρήστης επιλέγει στις ρυθμίσεις το «Inspect» και στην συνέχεια «Data».



Εικόνα 41 Grafana CSV

Στο νέο παράθυρο που θα εμφανιστεί διαλέγει τα «measurement» που θέλι να περιέχονται στο αρχείο και κατεβάζει το CSV.



Εικόνα 42 Grafana Download CSV

7 Αποτελέσματα Χρήσης Συστήματος

Στο συγκεκριμένο κεφάλαιο αναφέρονται τα αποτελέσματα, συμπεράσματα και οι προτάσεις μελλοντικής έρευνας πάνω στο σύστημα που υλοποιήθηκε κατά την διάρκεια της διπλωματικής εργασίας.

7.1 Αποτελέσματα

7.1.1 Αποτελέσματα Οπτικοποίησης και Ανάλυση Δεδομένων

Για την οπτικοποίηση και ανάλυση των δεδομένων χρησιμοποιείται μόνο η λειτουργικότητα που προσφέρεται από το λογισμικό Grafana. Όπως ήδη έχει αναφερθεί στο κεφάλαιο 6 το node-red δεν κρατάει τα δεδομένα στα αντίστοιχα διαγράμματα Chart που διαθέτει και αν γίνει επανεκκίνηση ο διακομιστής μηδενίζει όλες τις τιμές. Το συγκεκριμένο γεγονός γίνεται επειδή είναι ρυθμισμένο να οπτικοποιεί τα δεδομένα όπως τα λαμβάνει από τον MQTT Broker και όχι από την βάση InfluxDB.

Στις εικόνες που ακολουθούν παρουσιάζεται η οπτικοποίηση των δεδομένων που βρίσκονται στην βάση InfluxDB διαμέσου της εφαρμογής του Grafana. Στην κάθε εικόνα παρουσιάζονται διαφορετικές ενδείξεις διαφορετικής κατηγορίας αισθητήρων, προκειμένου να φανεί η σωστή λειτουργία τους. Επιπλέον για την κάθε εικόνα ορίζεται διαφορετικό διάστημα και χρονική περίοδος μετρήσεων, ώστε να φαίνεται η ικανότητα του Grafana στην ανάλυση δεδομένων, ανάλογα με τις επιθυμίες του χρήστη.

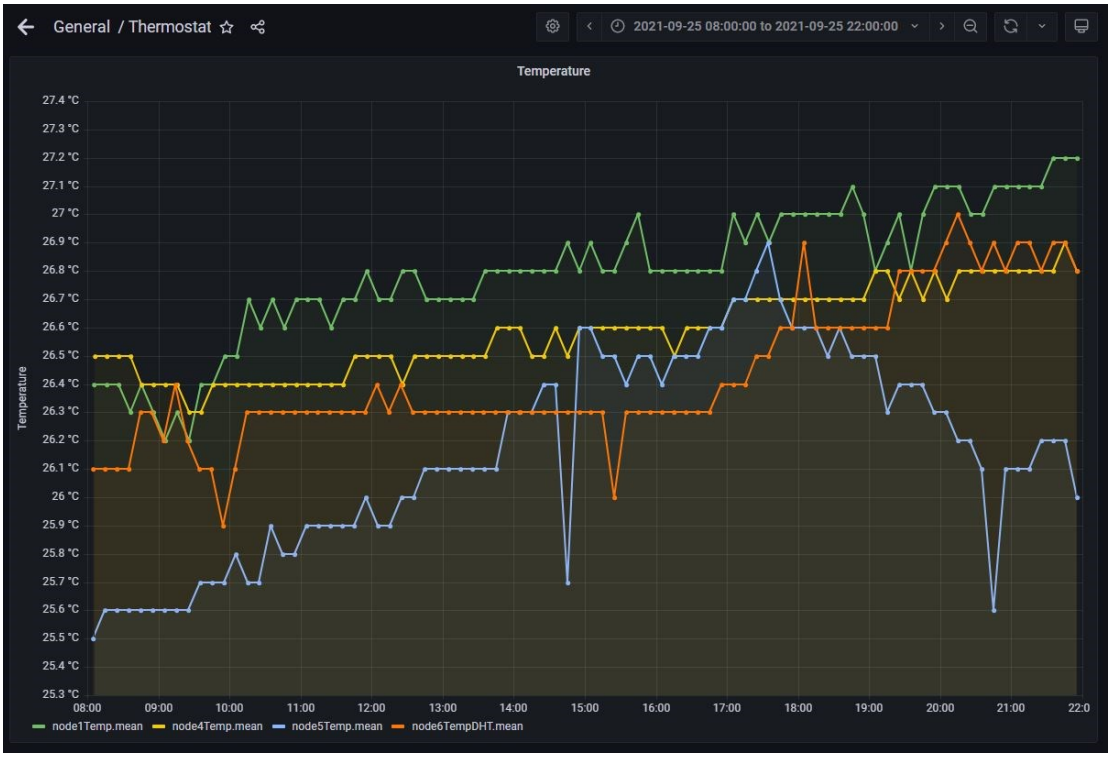
Στο γράφημα της εικόνας 43 φαίνεται η υγρασία όπως έχει καταγραφεί από όλες τις πλακέτες που διαθέτουν τον αισθητήρα DHT11. Όπως ήδη έχει αναφερθεί ο συγκεκριμένος αισθητήρας είναι υπεύθυνος για την καταμέτρηση της θερμοκρασίας και υγρασίας του χώρου που βρίσκεται η καθεμιά από αυτές τις πλακέτες ESP32.

Γίνεται δειγματοληψία κάθε δέκα λεπτά για την περίοδο 19/9/2021-13:00 έως 25/9/2021-22:00. Στο διάγραμμα φαίνονται το σύνολο των τιμών στην καταμέτρηση της υγρασίας σε διάρκεια 6 ημερών. Από τις γραμμές που περιέχονται στο διάγραμμα η κίτρινη και η πράσινη απευθύνονται στις πλακέτες εσωτερικού χώρου, η γαλάζια στην πλακέτα εξωτερικού χώρου και η πορτοκαλί στην πλακέτα του λεβητοστασίου.

Εικόνα 43 Θερμοκρασίες Αισθητήρων DHT11

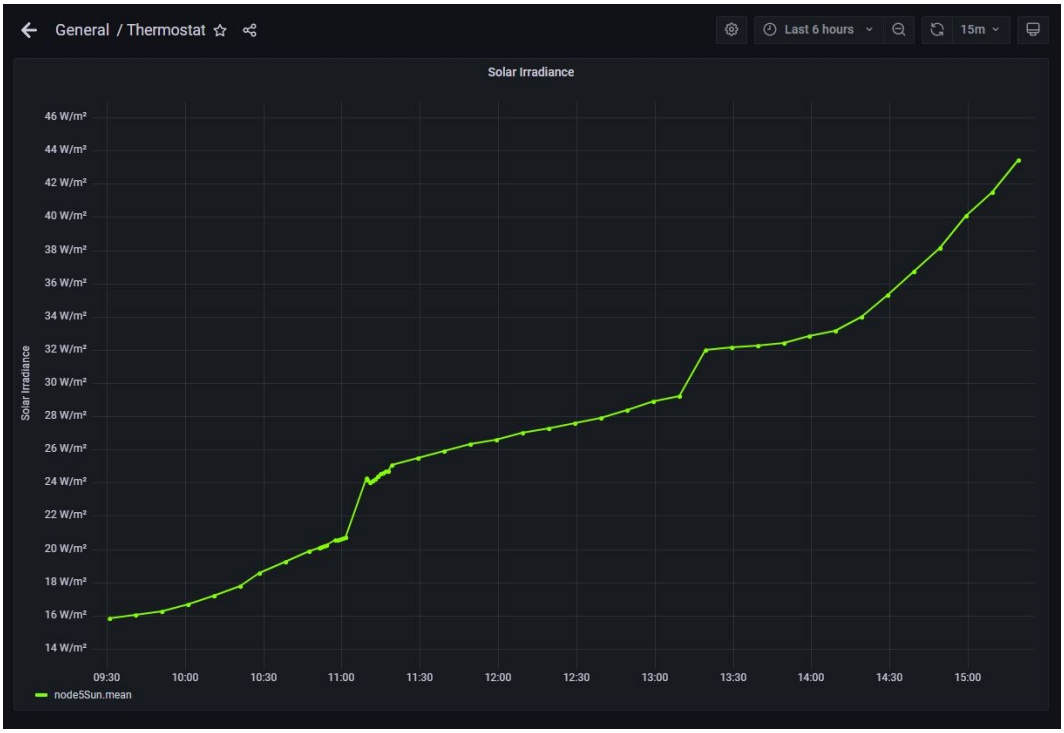
Στο γράφημα της εικόνας 44 φαίνεται η θερμοκρασία όπως έχει καταγραφεί από όλες τις πλακέτες που διαθέτουν τον αισθητήρα DHT11. Αντίθετα με πριν, στην συγκεκριμένη εικόνα ενδιαφερόμαστε για την ανάλυση μιας συγκεκριμένης μέρας από το πλήθος ημερών που παρουσιάστηκε στην προηγούμενη εικόνα. Παρατηρούμε ότι τα δεδομένα φαίνονται πιο ξεκάθαρα μιας και περιορίζουμε τον όγκο τους και στοχεύουμε σε αυτά που μας ενδιαφέρουν.

Γίνεται δειγματοληψία κάθε δέκα λεπτά για την περίοδο 25/9/2021-08:00 έως 25/9/2021-22:00. Από τις γραμμές που περιέχονται στο διάγραμμα η κίτρινη και η πράσινη απευθύνονται στις πλακέτες του υποσυστήματος εσωτερικού χώρου, η γαλάζια στην πλακέτα του υποσυστήματος εξωτερικού χώρου και η πορτοκαλί στην πλακέτα του λεβητοστασίου.



Εικόνα 44 Υγρασίες Αισθητήρων DHT11

Στην επόμενη εικόνα παρουσιάζεται η ηλιακή ακτινοβολία όπως έχει καταγραφεί και μετατραπεί από τον αισθητήρα BH1750. Στην συγκεκριμένη περίπτωση εξετάζουμε τις τελευταίες 6 ώρες της τρέχουσας μέρας. Ο συγκεκριμένος αισθητήρας είναι συνδεδεμένος με την πλακέτα του υποσυστήματος εξωτερικού χώρου.



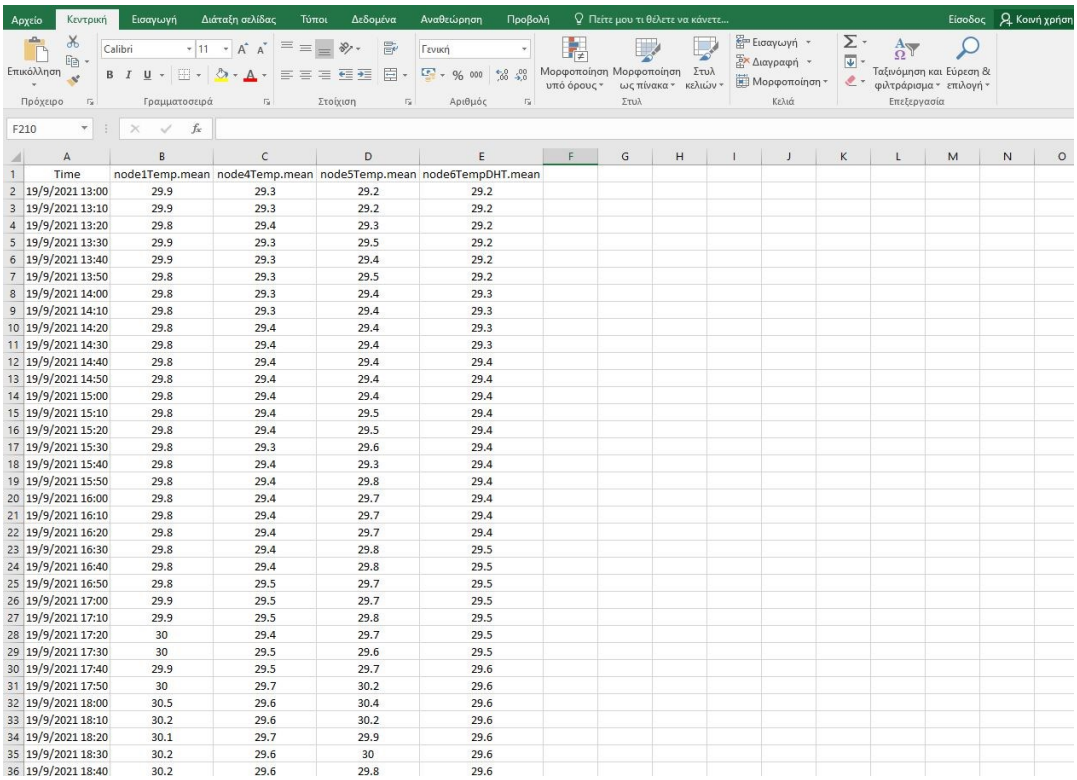
Εικόνα 7-3 Ηλιακή Ακτινοβολία

Στην εικόνα 45 παρουσιάζονται οι θερμοκρασίες όπως συλλέγονται από τους αισθητήρες επαφής DS18B20. Στην συγκεκριμένη περίπτωση χρησιμοποιούμε την ικανότητα της grafana να ορίζει SQL-Lite ερωτήματα για την οπτικοποίηση των δεδομένων. Όπως στην πρώτη εικόνα μελετάμε την περίοδο καταγραφής από το InfluxDB από 19/9/2021-13:00 έως 25/9/2021-22:00. Ωστόσο ομαδοποιούμε την στήλη του χρόνου μπαίνοντας στις ρυθμίσεις του γραφήματος και ορίζοντας το «GROUP BY time(3h)». Με την συγκεκριμένη εντολή το πεδίο με τα χρονικά στιγμιότυπα ομαδοποιείται και στο γράφημα εμφανίζεται η τελευταία τιμή που λήφθηκε για κάθε 3 ώρες για όλο το διάστημα των 6 ημερών του δείγματος. Από τις γραμμές που περιέχονται στο διάγραμμα η κίτρινη και η πράσινη απευθύνονται στις πλακέτες του υποσυστήματος εσωτερικού χώρου που είναι τοποθετημένες στο θερμαντικό σώμα και στον σωλήνα αντίστοιχα. Η πορτοκαλί απευθύνεται στην καταμέτρηση της πλακέτας του υποσυστήματος του λεβητοστασίου.

Εικόνα 45 Θερμοκρασίες Αισθητήρων DS18B20

Τέλος όπως ήδη έχουμε αναφέρει το Grafana μας δίνει την ιδιότητα να εξάγουμε EXCEL CSV. Το συγκεκριμένο excel εξάγεται με βάση τους πίνακες που έχουμε δημιουργήσει οπότε όπως είναι φυσικό καθορίζεται και σε αυτό το ίδιο χρονικό διάστημα που έχει οριστεί να οπτικοποιήσει το Grafana. Στην εικονα 46 παρουσιάζεται το excel για την

θερμοκρασία των αισθητήρων DHT11 από τις από 19/9/2021-13:00 έως 25/9/2021-22:00. Οι στήλες node1Temp και node4 Temp απευθύνονται σε πλακέτες από το υποσύστημα του εσωτερικού χώρου, η στήλη με το όνομα node5Temp απευθύνεται στην πλακέτα του υποσυστήματος του εξωτερικού χώρου και η node6TempDHT στην πλακέτα του συστήματος του λεβητοστασίου.



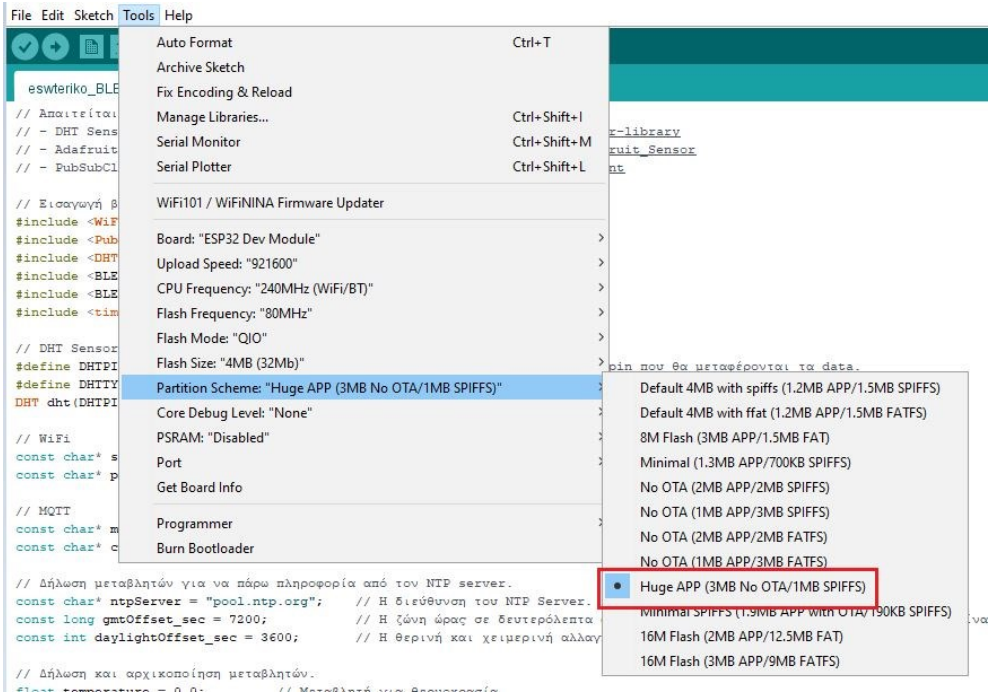
Time	node1Temp.mean	node4Temp.mean	node5Temp.mean	node6TempDHT.mean
19/9/2021 13:00	29.9	29.3	29.2	29.2
19/9/2021 13:10	29.9	29.3	29.2	29.2
19/9/2021 13:20	29.8	29.4	29.3	29.2
19/9/2021 13:30	29.9	29.3	29.5	29.2
19/9/2021 13:40	29.9	29.3	29.4	29.2
19/9/2021 13:50	29.8	29.3	29.5	29.2
19/9/2021 14:00	29.8	29.3	29.4	29.3
19/9/2021 14:10	29.8	29.3	29.4	29.3
19/9/2021 14:20	29.8	29.4	29.4	29.3
19/9/2021 14:30	29.8	29.4	29.4	29.3
19/9/2021 14:40	29.8	29.4	29.4	29.4
19/9/2021 14:50	29.8	29.4	29.4	29.4
19/9/2021 15:00	29.8	29.4	29.4	29.4
19/9/2021 15:10	29.8	29.4	29.5	29.4
19/9/2021 15:20	29.8	29.4	29.5	29.4
19/9/2021 15:30	29.8	29.3	29.6	29.4
19/9/2021 15:40	29.8	29.4	29.3	29.4
19/9/2021 15:50	29.8	29.4	29.8	29.4
19/9/2021 16:00	29.8	29.4	29.7	29.4
19/9/2021 16:10	29.8	29.4	29.7	29.4
19/9/2021 16:20	29.8	29.4	29.7	29.4
19/9/2021 16:30	29.8	29.4	29.8	29.5
19/9/2021 16:40	29.8	29.4	29.8	29.5
19/9/2021 16:50	29.8	29.5	29.7	29.5
19/9/2021 17:00	29.9	29.5	29.7	29.5
19/9/2021 17:10	29.9	29.5	29.8	29.5
19/9/2021 17:20	30	29.4	29.7	29.5
19/9/2021 17:30	30	29.5	29.6	29.5
19/9/2021 17:40	29.9	29.5	29.7	29.6
19/9/2021 17:50	30	29.7	30.2	29.6
19/9/2021 18:00	30.5	29.6	30.4	29.6
19/9/2021 18:10	30.2	29.6	30.2	29.6
19/9/2021 18:20	30.1	29.7	29.9	29.6
19/9/2021 18:30	30.2	29.6	30	29.6
19/9/2021 18:40	30.2	29.6	29.8	29.6

Εικόνα 46 Θερμοκρασίες DHT11 CSV

7.1.2 Λειτουργικότητας συστήματος

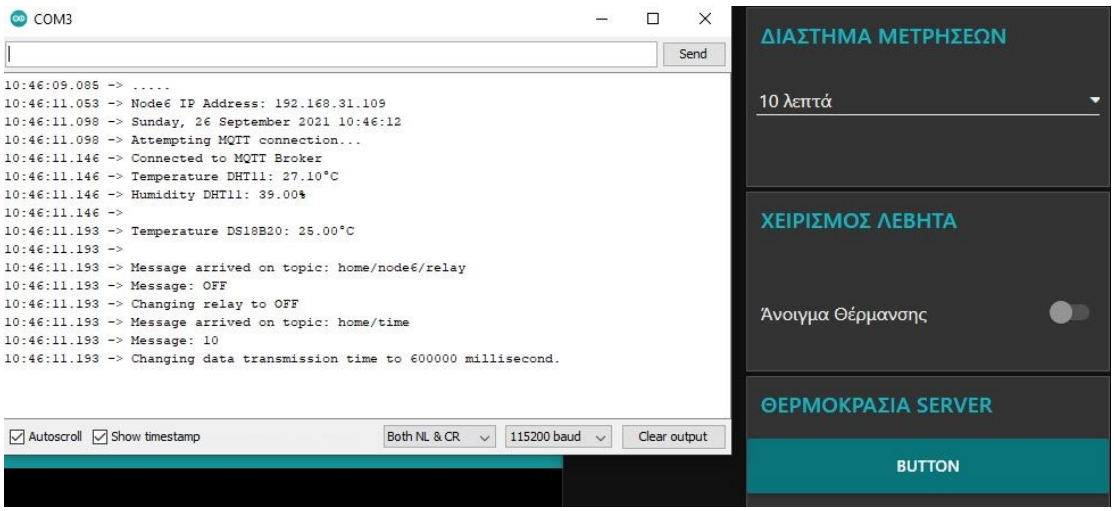
Στην συγκεκριμένη ενότητα θα παρουσιαστεί η λειτουργικότητα των πλακετών με βάση των προγραμματιστικών ιδιοτήτων που τις δόθηκαν.

Αρχικά πρέπει να αναφερθεί ότι ένα πρόβλημα που αντιμετωπίστηκε κατά την υλοποίηση είναι το μεγάλο μέγεθος της βιβλιοθήκης για το πρωτόκολλο BLE. Η συγκεκριμένη βιβλιοθήκη δεν μπορεί να συνυπάρξει με την αντίστοιχη του WiFi και του πρωτοκόλλου MQTT. Αυτό έχει ως αποτέλεσμα να βγάζει σφάλμα κατά την προσπάθεια μεταγλώττισης του προγράμματος που απευθύνεται στην πλακέτα gateway. Το πρόβλημα λύνεται αν κατά την διάρκεια της μεταγλώττισης αυξηθεί η μνήμη που διατίθεται στην εφαρμογή επιλέγοντας την επιλογή «Huge APP» όπως φαίνεται στην εικόνα 47.



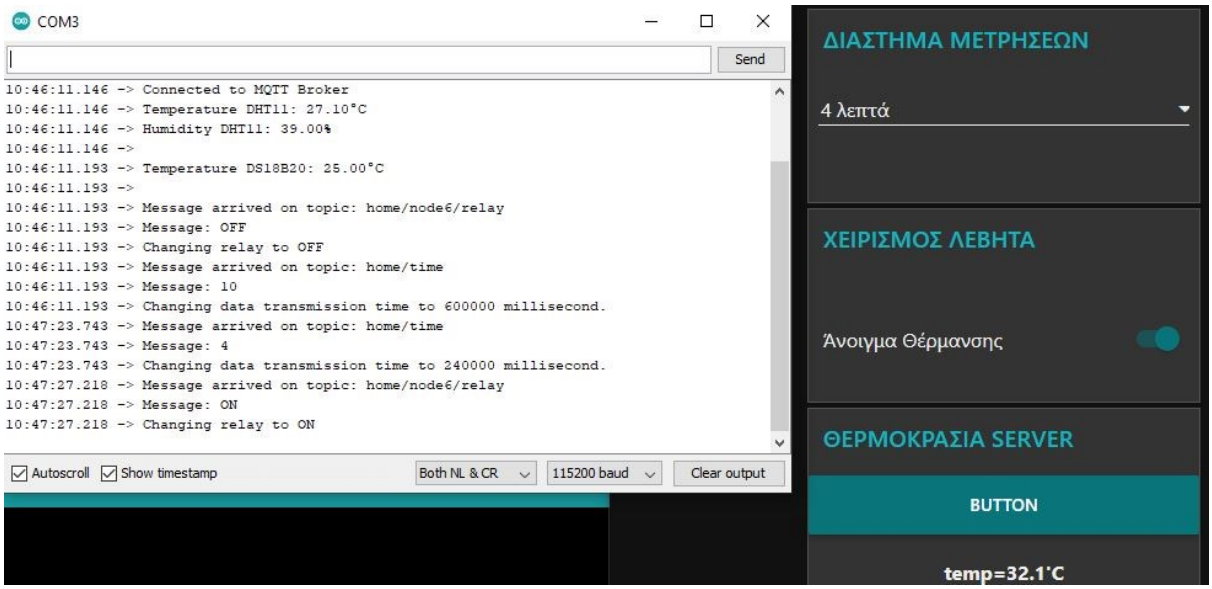
Εικόνα 47 Huge APP

Στην εικόνα 48 παρουσιάζεται η δυνατότητα ρύθμισης του πρωτοκόλλου MQTT ώστε να τεθεί ένα θέμα ως «διατηρούμενο μήνυμα», δηλαδή να έχει την ικανότητα να διατηρεί την τελευταία τιμή που έλαβε με αποτέλεσμα να την λαμβάνει κάθε νέος Client που συνδέεται. Η συγκεκριμένη ιδιότητα δόθηκε στα θέματα «home/time» και «home/node6/relay» διαμέσου του node-red. Συνδέθηκε στον υπολογιστή η πλακέτα που υλοποιεί το σύστημα του λεβητοστασίου και παρατηρήθηκε ότι απευθείας μετά την σύνδεση λαμβάνει τις τελευταίες τιμές των αντίστοιχων θεμάτων.



Εικόνα 48 Διατηρούμενο Μήνυμα

Στην συνέχεια από το node-red δόθηκαν εντολές ως χρήστης για τον καθορισμό νέου χρονικού διαστήματος ορισμένου στα 4 λεπτά. Επίσης θέσαμε σε λειτουργία την θέρμανση. Στην εικόνα 49 παρατηρείται ότι λήφθηκαν τα αντίστοιχα μηνύματα και καθορίστηκε η νέα λειτουργικότητα στις πλακέτες ESP32.



Εικόνα 49 Εντολές Χρήστη Με Node-Red

Επιπλέον στον κώδικα που αναπτύχθηκε δημιουργήθηκε προστασία σε περίπτωση απώλειας σύνδεσης WiFi ή σε περίπτωση αδυναμία σύνδεσης με τον MQTT Broker. Δηλαδή όταν για κάποιο λόγο χάνεται μία από τις δύο συνδέσεις, τότε αυτομάτως οι πλακέτες κάνουν ενέργειες επανασύνδεσης. Για το πρωτόκολλο MQTT ακολουθήθηκε ένα από τα παραδείγματα της αντίστοιχης βιβλιοθήκης που στην αρχή της συνάρτησης loop() ελέγχει σε κάθε επανάληψη αν η πλακέτα δεν είναι συνδεδεμένη με τον Broker. Όταν η συνθήκη είναι αληθής προβαίνει σε ενέργειες επανασύνδεσης.

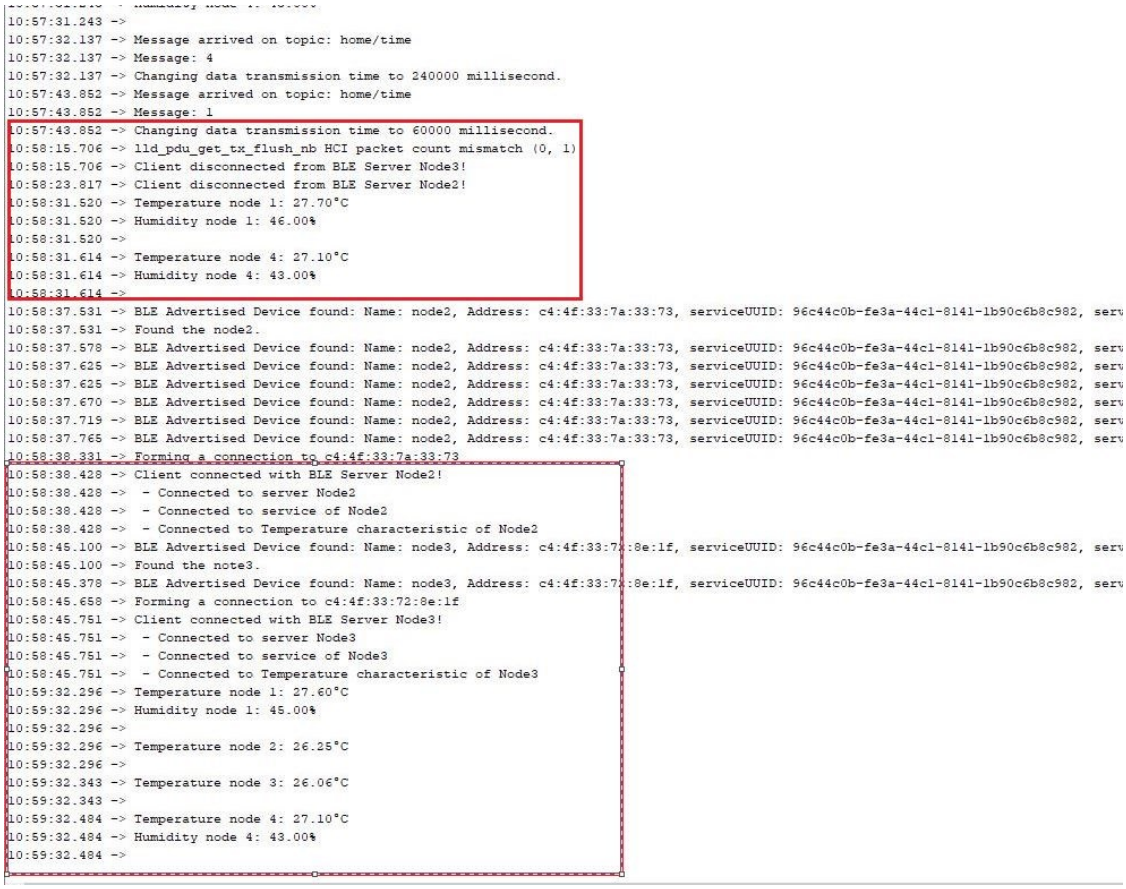
Για την σύνδεση WiFi χρησιμοποιήθηκε μια από τις ιδιότητες του ESP32 καθορίσθηκε ένα από τα λεγόμενα «Event». Γράφοντας την εντολή «WiFi.onEvent(wifiDisconnected, SYSTEM_EVENT_STA_DISCONNECTED);» στην συνάρτηση της Setup() ορίζεται η συνάρτηση wifiDisconnect, ως συνάρτηση εκτέλεσης του event. Η wifiDisconnect έχει προγραμματιστεί να εκτυπώνει απλώς τον λόγο της αποσύνδεσης, ωστόσο από την στιγμή που ενεργοποιείται το Event αυτόματα το ESP32 προβαίνει σε ενέργειες επανασύνδεσης με τον δρομολογητή του WiFi. Όπως φαίνεται στην επόμενη εικόνα έγινε επανεκκίνηση στον δρομολογητή ενώ είναι συνδεδεμένη μια πλακέτα στον υπολογιστή. Παρατηρούνται

συνεχόμενα μηνύματα απώλειας σύνδεσης αν και το event δεν είναι γραμμένο σε κάποια εντολή επανάληψης, ωστόσο κάνει συνεχόμενους ελέγχους επανασύνδεσης. Το πλεονέκτημα αυτής της μεθόδου είναι ότι τα events τρέχουν σε παράλληλη ροή από αυτήν του υπόλοιπου κώδικα, δεν επηρεάζουν δηλαδή την εκτέλεση του κώδικα και αυτό φαίνεται ότι ταυτόχρονα εκτυπώνονται μηνύματα απώλειας του MQTT Broker. (Απώλεια της σύνδεσης WiFi συνεπάγεται και απώλεια της σύνδεσης με τον MQTT Broker). Τέλος όταν εντοπιστεί ο δρομολογητής επανέρχεται η σύνδεση και η πλακέτα λειτουργεί κανονικά.

Εικόνα 50 Επανασύνδεση WiFi

Τέλος μια σημαντική λειτουργικότητα που δόθηκε στο υποσύστημα του εσωτερικού χώρου είναι η διαδικασία επανασύνδεσης σε περίπτωση που διακοπεί η σύνδεση μεταξύ των πλακετών που το αποτελούν χρησιμοποιώντας το πρωτόκολλο Bluetooth Low Energy. Όπως ήδη έχει αναφερθεί στο κεφάλαιο 3 ένας BLE Client μπορεί να συνδεθεί με έναν BLE Server μόνο αν ο Server διαφημίζει τον εαυτό του. Οπότε στον κώδικα που υλοποιείται από τον BLE Server στην κλάση «class MyServerCallbacks: public BLEServerCallbacks» ορίσαμε ότι αν λάβει κάποια αίτηση αποσύνδεσης να προβαίνει αυτόματα σε διαδικασία διαφήμισης. Επιπλέον στον BLE Client στην αρχή της συνάρτησης loop() δημιουργήθηκε μια συνθήκη «if» με την οποία καλείται η συνάρτηση «scan» σε περίπτωση που έστω και μια πλακέτα από τους BLE Server χαθεί από την σύνδεση. Όπως φαίνεται στην εικόνα 51, οι πλακέτες με ονόματα «node2» και «node3» απενεργοποιούνται. Αυτό έχει ως

αποτέλεσμα να αποσυνδεθούν από τον BLE Client που εκτελεί χρέη gateway. Όταν ο χρήστης επαναφέρει την λειτουργία τους τότε ο Client επικαλείται την διαδικασία επανασύνδεσης και συνδέεται μαζί τους.



Εικόνα 51 BLE Επανασύνδεση

Με βάση την συγκεκριμένη εικόνα παρατηρούμε και μια επιπλέον λειτουργικότητα του προγράμματος εκτέλεσης. Όλες οι πλακέτες από όλα τα υποσυστήματα είναι ορισμένες ώστε να μην στέλνονται δεδομένα στον MQTT Broker σε περίπτωση αδυναμίας διαβάσματος των αισθητήρων που κατέχουν. Επίσης η πλακέτα gateway του υποσυστήματος εσωτερικού χώρου έχει ρυθμιστεί να μην στέλνει μήνυμα στον MQTT Broker εκ μέρους των BLE Server που έχουν αποσυνδεθεί. Όπως φαίνεται στην παραπάνω εικόνα όταν χάθηκε η σύνδεση από τα «node2» και «node3» στάλθηκαν μηνύματα μόνο για τους υπόλοιπους αισθητήρες. Αν δεν είχε γίνει ο συγκεκριμένος έλεγχος το MQTT πρωτόκολλο θα έστελνε κενά μηνύματα και το InfluxDB θα τα αντικαθιστούσε με μηδενικά, με αποτέλεσμα να χαλούσε η ροή στα δεδομένα που συλλέγει.

7.2 Συμπεράσματα

Αν και η συγκεκριμένη διπλωματική εργασία δεν υλοποιήθηκε σε λειτουργικό σύστημα θέρμανσης, λόγο της περιόδου που μελετήθηκε, ωστόσο ο τρόπος συλλογής των δεδομένων και η ανάλυσή τους δεν θα είχε διαφορές από ένα πλήρες λειτουργικό σύστημα. Οι υλοποιήσεις παρόμοιων συστημάτων σε κτήρια και οικιακούς χώρους, καθώς και η μελέτη των δεδομένων που συλλέγουν είναι ένας τομέας που μελετάτε πολύ τα τελευταία χρόνια εξαιτίας των πλεονεκτημάτων που προσφέρει. Το σύστημα που μελετήθηκε και παρουσιάζεται, όχι μόνο αναβαθμίζει την ποιότητας ζωής των χρηστών, αλλά έχει την δυνατότητα να συμβάλλει σε ένα από τα μεγαλύτερα προβλήματα που ταλαιπωρούν τον πλανήτη μας, αυτό της κλιματικής αλλαγής.

Στην παρούσα εργασία χρησιμοποιήθηκαν πλακέτες που διαθέτουν τον μικρολεγκτή ESP32 και τροφοδοτήθηκαν με μπαταρία προκειμένου να τοποθετηθούν σε οποιαδήποτε περιβάλλον χωρίς να είναι απαραίτητη η καλωδίωσή τους. Η χρήση των πρωτοκόλλων επικοινωνίας BLE και MQTT δημιούργησαν ένα αποδοτικό σύστημα ανταλλαγής μηνυμάτων και συλλογής δεδομένων. Η χρησιμοποίηση του raspberry pi 4 ως διακομιστή απέδειξε ότι ακόμα και σε τοπικό επίπεδο με ελάχιστα έξοδα μπορεί κάποιος να δημιουργήσει ένα αποδοτικό σύστημα.

Τέλος όσο αφορά την επιλογή των λογισμικών αποθήκευσης και οπτικοποίησης δεδομένων, δηλαδή του Mosquitto Broker, του Node-Red, του InfluxDB και του Grafana, στην αρχή φάνηκαν δυσνόητα ως προς την χρήση τους. Ωστόσο μέσα στην πρώτη βδομάδα ο οποιοσδήποτε μπορεί να εξοικειωθεί με αυτά διαβάζοντας απλώς τις τεκμηριώσεις λειτουργικότητας (Documedation) που προσφέρει η αντίστοιχη εταιρία και προβαίνοντας στις αντίστοιχες δοκιμές. Επιπλέον ένα μεγάλο πλεονέκτημα είναι ότι τα συγκεκριμένα λογισμικά συνεργάζονται τέλεια μεταξύ τους και υπάρχει ένα μεγάλο κοινό που τα χρησιμοποιεί και τα υποστηρίζει, με αποτέλεσμα να βρίσκει κάποιος εύκολα λύσεις σε ότι πρόβλημα αντιμετωπίζει. Από την χρήση του Node-Red καταλαβαίνει κάποιος ότι η ανάλυση δεδομένων διαμέσου του dashboard που προσφέρει, δεν είναι ένα από τα δυνατά του σημεία. Ωστόσο διαμέσου του συγκεκριμένου λογισμικού μπορεί να δημιουργηθεί μια υλοποίηση IoT που σε αντίθετη περίπτωση θα απαιτούσε πολλές προγραμματιστικές ώρες και πολλές γνώσεις προγραμματισμού. Αυτό κάνει το Node-Red

ιδανικό στο να δημιουργείς εύκολα και γρήγορα περιβάλλοντα διεπαφής χρήστης για τον χειρισμό υλοποιήσεων IoT. Αντίθετα το Grafana είναι ιδανικό στο να αναλύεις και να οπτικοποιείς μεγάλες συλλογές δεδομένων. Όπως φάνηκε και στην προηγούμενη ενότητα προσφέρει στον χρήστη μια πληθώρα εργαλείων και δημιουργεί ένα πολύ όμορφο οπτικό αποτέλεσμα. Επίσης δίνεται η δυνατότητα να χειρίζεται κάποιος πολλές διαφορετικές βάσεις και να ορίζει πολλές διαφορετικές διεπαφές χρήστη.

7.2.1 Συμπεράσματα και προβλήματα κατά την υλοποίηση

Όσο αφορά τα υλικά που χρησιμοποιήθηκαν στο συγκεκριμένο σύστημα όλα λειτουργούσαν ικανοποιητικά με βάση τις προδιαγραφές που διαθέτουν. Ένα σημαντικό πρόβλημα που παρουσιάστηκε την περίοδο του καλοκαιριού ήταν η πολύ μεγάλη θερμοκρασία της CPU του Raspberry Pi 4. Αν και αρχικά χρησιμοποιήθηκαν απλές μικρές ψήκτρες (Heatsink) και ένας μικρός ανεμιστήρας, ωστόσο υπήρχε η ανάγκη αγοράς ενός πιο αποδοτικού μέσου. Αγοράστηκε από το amazon μια ψύχρα με την ονομασία «Ice Tower» εικόνα της οποίας υπάρχει στο κεφάλαιο 5.

Τέλος όσο αφορά την υλοποίηση του κώδικα όλες οι βιβλιοθήκες που χρησιμοποιήθηκαν εκτός της BLE ήταν φιλικές προς τον χρήστη. Τόσο η βιβλιοθήκη <WiFi.h> υπεύθυνη για την σύνδεση WiFi όσο και η βιβλιοθήκη <PubSubClient.h> υπεύθυνη για την σύνδεση MQTT λειτουργούσαν τέλεια, καθώς και οι αντίστοιχες των αισθητήρων. Οι συγκεκριμένες βιβλιοθήκες ήταν εύκολες στην χρήση και διέθεταν προστασία σφαλμάτων διαμέσου την επιστρεφόμενη τιμής στις συναρτήσεις που καλούσες. Αντίθετα η βιβλιοθήκη BLE ήταν πιο δύσχρηστη και πολλές από τις συναρτήσεις που χρησιμοποιείς δεν επιστρέφουν κάποια τιμή με αποτέλεσμα να σταματάει απλώς το πρόγραμμα ή να κάνει επανεκκίνηση η πλακέτα ESP32 αν κάνεις εσφαλμένη χρήση τους. Το συγκεκριμένο γεγονός οδηγεί στην ανάγκη δημιουργίας προστασίας με την χρήση σημαιών (flag), οπότε για ένα σύστημα gateway αυτό δημιουργεί την ανάγκη πολλών πρόσθετων μεταβλητών. Τέλος μια πρόσθετη δυσκολία στην υλοποίηση του προγραμματισμού είναι ότι ενώ για όλες τις βιβλιοθήκες χρησιμοποιήθηκε η C, για την βιβλιοθήκη BLE έπρεπε οι εντολές να είναι σε C++ . Αν και οι δυο γλώσσες είναι

παραπλήσιες, στην υλοποίηση δημιουργείται μια δυσκολία κυρίως όσο αφορά την χρήση αντικειμένων και string.

7.3 Μελλοντική Έρευνα

Στην ενότητα αυτή πρέπει να αναφερθούν οι μελλοντικές επεκτάσεις της εργασίας, καθώς από τη φύση της η εργασία μπορεί να υποστηρίξει επεκτάσεις και βελτιώσεις σε πολλά επίπεδα.

Ένα πρώτο σενάριο είναι ότι θα μπορούσε να επεκταθεί το υποσύστημα του λεβητοστασίου προσθέτοντας αισθητήρα και relay για τον χειρισμό του θερμοσίφωνα και επιπλέον την καταμέτρηση της στάθμης του πετρελαίου.

Στη συνέχεια, προτείνεται η επέκτασή του συστήματος στο να περιλαμβάνει κλιματιστικό χώρου καθώς και αφυγραντήρα. Αυτό θα έχει ως αποτέλεσμα τον πλήρη έλεγχο του περιβάλλοντος του κτιρίου.

Ένα σημείο που χρήζει βελτίωσης έγκειται στην σύνδεση πάνελ στην πλακέτα του εξωτερικού χώρου. Με αυτό τον τρόπο δεν θα χρειαζόταν ανθρώπινη επέμβαση, όσο αφορά το θέμα της φόρτισης της μπαταρίας που διαθέτει.

Μια επιπλέον βελτίωση είναι η προσθήκη πρόσθετων αυτοματοποιημένων λειτουργιών στο σύστημα τα οποία όταν οι αισθητήρες έφταναν σε μια συγκεκριμένη τιμή τότε θα ενεργοποιούταν η κινητικότητά τους. Παραδείγματος χάρη όταν η ηλικιακή ακτινοβολία έφτανε σε μια προκαθορισμένη τιμή και ήταν καλοκαίρι, τότε αυτόματα θα κατέβαιναν τα παντζούρια του παραθύρου για να προστατευτεί ο εσωτερικός χώρος.

Ένα σημαντικό σημείο της υλοποίησης, που δεν πραγματοποιήθηκε λόγω περιορισμένου χρόνου, είναι η δημιουργία μια εφαρμογής σε κινητό Android χρησιμοποιώντας το Android Studio ή μια παρεμφερή εφαρμογή. Αν και τα λογισμικά που χρησιμοποιήθηκαν προσαρμόζονται στην οθόνη της συσκευής, μια εφαρμογή Android θα έδινε καλύτερο αποτέλεσμα και πιο εύκολη χρήση.

Τέλος μια σημαντική επέκταση του συστήματος είναι να προσαρμοστεί το σύστημα επικοινωνίας των μικροελεγκτών σε mesh δίκτυο χρησιμοποιώντας το πρωτόκολλο

επικοινωνίας Bluetooth Low Energy. Ένα τέτοιο σύστημα θα μπορούσε να αυξήσει την κάλυψη του σημείου πρόσβασης που έθετε η πλακέτα gateway, μιας και οι κόμβοι δεν χρειάζεται να συνδεθούν με έναν κεντρικό κόμβο. Οι πλακέτες μπορούν να αυτό-οργανωθούν και να μιλήσουν δυναμικά μεταξύ τους και αν κάποιος κόμβος αφαιρεθεί από το δίκτυο η θέση του θα αναπληρωνόταν από τους υπόλοιπους.

Βιβλιογραφία

[1] IoT Heating Solutions For The Future [Ηλεκτρονικό]. Available: https://www.danfoss.com/en/about-danfoss/articles/dhs/iot-heating-solutions-for-the-future/

[2] Internet of Things (IoT) [Ηλεκτρονικό]. Available: https://www.tutorialspoint.com/internet_of_things/index.htm

[3] Digiesi, S.; Mossa, G.; Mummolo, G. Supply lead time uncertainty in a sustainable order quantity inventory model. Manag. Prod. Eng. Rev. 2013, 4, 15–27.

[4] Facchini, F.; Mummolo, G.; Mossa, G.; Digiesi, S.; Boenzi, F.; Verriello, R. Minimizing the carbon footprint of material handling equipment: Comparison of electric and LPG forklifts. J. Ind. Eng. Manag. 2016, 9, 1035–1046.

[5] Ejaz, W.; Naeem, M.; Shahid, A.; Anpalagan, A.; Jo, M. Efficient energy management for the internet of things in smart cities. IEEE Commun. Mag. 2017, 55, 84–91.

[6] Jouhara, H.; Yang, J. Energy efficient HVAC systems. Energy Build. 2018, 179, 83–85.

[7] K. Pahlavan, P. Krishnamurthy, (2020) “Evolution and Impact of Wi-Fi Technology and Applications: A Historical Perspective”. Available: https://link.springer.com/content/pdf/10.1007/s10776-020-00501-8.pdf

[8] Bluetooth [Ηλεκτρονικό]. Available: https://www.bluetooth.com/learn-about-bluetooth/tech-overview/

[9] A. Kevin Townsend, Carles Cufi and R. Davidsoni, Getting Started with Bleutooth Low Energy,TOOLS AND TECHNIQUES FOR LOW-POWER NETWORKING, 1st ed. O’REILLY, May 2014.

[10] BLE C++ [Ηλεκτρονικό]. Available: https://github.com/nkolban/esp32-snippets/blob/master/Documentation/BLE%20C%2B%2B%20Guide.pdf

[11] Bluetooth Assigned Numbers [Ηλεκτρονικό]. Available: https://www.bluetooth.com/specifications/assigned-numbers/

[12] OASIS, «MQTT Version 3.1.1 - OASIS Standard,» Οκτώβριος 2014. [Ηλεκτρονικό]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

[13] Tracy, P., "MQTT protocol minimizes network bandwidth for the internet of things "Νοέμβριος 2016. [Ηλεκτρονικό]. Available: https://www.rcrwireless.com/20161129/fundamentals/mqtt-internet-of-things-tag31-tag99

[14] Quality of Service 0,1 & 2 - MQTT Essentials: Part 6. [Ηλεκτρονικό]. Available: https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/

[15] ESP32 WROOM32 Datasheet. [Ηλεκτρονικό]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf

[16] N. Kolban, Kolban's Book on ESP32, 2018. [Ηλεκτρονικό]. Available: https://leanpub.com/kolban-ESP32

[17] Analog to Digital Converter (ADC) [Ηλεκτρονικό]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/adc.html

[18] Cheddadi, Y.; (2020), "Design and implementation of an intelligent low-cost IoT solution for energy monitoring of photovoltaic stations" [Ηλεκτρονικό]. Available: https://link.springer.com/article/10.1007/s42452-020-2997-4

[19] Michael P.; Johnston D.; Moreno W.; (2020), "A conversion guide: solar irradiance and lux illuminance" [Ηλεκτρονικό]. Available: https://www.jvejournals.com/article/21667

[20] Raspberry Pi [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Raspberry_Pi

[21] Raspberry Pi 4 Tech Specs [Ηλεκτρονικό]. Available: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/

[22] Raspberry Pi Documentation [Ηλεκτρονικό]. Available: https://www.raspberrypi.org/documentation/raspbian/

[23] Eclipse Mosquitto [Ηλεκτρονικό]. Available: https://mosquitto.org/

[24] Node-RED: Lecture 1 – A brief introduction to Node-RED [Ηλεκτρονικό]. Available: http://noderedguide.com/nr-lecture-1/

[25] Chaczko, Z., & Braun, R. (2017, July). Learning data engineering: Creating IoT apps using the node-RED and the RPI technologies. In Information Technology Based Higher Education and Training (ITHET), 2017 16th International Conference on (pp. 1-8). IEEE.

[26] Node-Red Flows [Ηλεκτρονικό]. Available: https://nodered.org/docs/user-guide/editor/workspace/flows

[27] Get started with InfluxDB Ηλεκτρονικό]. Available: https://docs.influxdata.com/influxdb/v2.0/get-started/

[28] Components of the TICK Stack [Ηλεκτρονικό]. Available: https://www.influxdata.com/time-series-platform/

[29] Grafana Wikipedia [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Grafana

[30] Grafana [Ηλεκτρονικό]. Available: https://grafana.com/

[31] The open-source platform for monitoring and observability [Ηλεκτρονικό]. Available: https://github.com/grafana/grafana

[32] Carli R.; (2020), "IoT Based Architecture for Model Predictive Control of HVAC Systems in Smart Buildings. Manag", pp. 1-18.

[33] Joseph M.; Jose V.; Habeeb A.; (2015), "Thermal Performance of Buildings: Case Study and Experimental Validation of Educational Building", pp. 4968-4974.

[34] Mangematin E.; Guillaume Pandraud G.; Roux D.; (2012) "Quick measurements of energy efficiency of buildings", pp. 383-390.

[35] Mangematin E.; Guillaume Pandraud G.; Roux D.; (2020) "Energy Efficiency in Smart Buildings: IoT Approaches", pp. 63679-63699.

[36] MQTT API Documentation [Ηλεκτρονικό]. Available: https://pubsubclient.knolleary.net/api

[37] Grafana documentation [Ηλεκτρονικό]. Available: https://grafana.com/docs/grafana/latest/visualizations/

Παράρτημα Α: ΚΩΔΙΚΑΣ ΣΥΣΤΗΜΑΤΟΣ ΘΕΡΜΟΣΤΑΤΗ

1 Υποσύστημα Εσωτερικού Χώρου

1.1 Πλακέτες που έχουν τον ρόλο του BLE Server (Node 2, Node 3, Node 4):

```
// Απαιτείται εγκατάσταση των ακόλουθων βιβλιοθήκων:
// - OneWire από Paul Stoffregen: https://github.com/PaulStoffregen/OneWire
// - DallasTemperature από Miles Burton: https://github.com/milesburton/Arduino-Temperature-Control-Library
// - DHT Sensor Library από Adafruit: https://github.com/adafruit/DHT-sensor-library
// - Adafruit Unified Sensor από Adafruit: https://github.com/adafruit/Adafruit_Sensor

// Βιβλιοθήκες για το BLE.
#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEServer.h>

// Αναγνωριστικά που ξεκλειδώνουν τα αντίστοιχα block κώδικα ανάλογα με τον
// αισθητήρα που έχει η κάθε πλακέτα.
#define DS18B20
//#define DHT11
#define BOARD_NAME "node3"       //Το όνομα του BLE Server.

// Αρχικοποίηση για αισθητήρα DS18B20.
#ifdef DS18B20
#include <OneWire.h>                // Βιβλιοθήκες για τον αισθητήρα DS18B20.
#include <DallasTemperature.h>
const int oneWireBus = 33;          // Ορίζω μια μεταβλητή με τον αριθμό του pin.
OneWire oneWire(oneWireBus);
DallasTemperature sensors(&oneWire);
#endif


// Αρχικοποίηση για τον αισθητήρα DHT11.
```

```
#ifdef DHT11
#include <DHT.h>              // Βιβλιοθήκη για τον αισθητήρα DHT11.
#define DHTPIN 33             // Ορίζω μια σταθερή μεταβλητή με τον αριθμό του pin.
#define DHTTYPE DHT11         // Οριζω μια σταθερή μεταβλητή με τον τύπο DHT.
DHT dht(DHTPIN,DHTTYPE);  // Αρχικοποίηση του αισθητήρα DHT11.
#endif

// Μοναδικά αναγνωριστικά UUID.
#define SERVICE_UUID "96c44c0b-fe3a-44c1-8141-1b90c6b8c982"
#define CHAR_TEMP_UUID "5578ca4c-db3e-4d14-86ac-287507abade6"
#define CHAR_HUM_UUID "856f5b18-7c3d-4eb3-a124-c4aa72ce6892"

// Δήλωση και αρχικοποίηση μεταβλητών.
BLECharacteristic* pCharacTemp;   // Χαρακτηριστικό με την θερμοκρασία.
BLEServer* pServer;
float temp = 0.0;                 // Μεταβλητή για θερμοκρασία.
char tempString[8];
// Σημαία για το αν είναι συνδεδεμένη η συσκευή. Μόνο όταν συνδεθεί θα χρησιμοποιεί
//τον αισθητήρα και θα στέλνει τις μετρήσεις.
bool devConnected = false;

#ifdef DHT11
BLECharacteristic* pCharacHum;   // Χαρακτηριστικό με την υγρασία.
float hum = 0.0;                          // Μεταβλητή για υγρασία.
char humString[8];
#endif

// Δήλωση πρόσθετων συναρτήσεων.
void advertising();             // Συνάρτηση που είναι υπεύθυνη για την διαφήμιση.


//------------------------------CLASSES-------------------------------
class MyServerCallbacks: public BLEServerCallbacks {
   void onConnect(BLEServer* pServer) {                  // Όταν γίνεται σύνδεση.
     devConnected = true;                                // Όταν συνδεθεί κάνω την σημαία true.

     Serial.println("Connect with BLE client!");
```

```
  }
   void onDisconnect(BLEServer* pServer) {              // Όταν γίνεται αποσύνδεση.
    devConnected = false;                          // Όταν αποσυνδεθεί κάνω την σημαία false.
    Serial.println("Disconnect from BLE client!");
    advertising();                      // Ξαναρχίζει η διαφήμιση μόλις διακοπεί η σύνδεση.
   }
}; // MyServerCallbacks


class MyCallbacks: public BLECharacteristicCallbacks {
   void onRead(BLECharacteristic* pCharacteristic) {
      Serial.print("Characteristic UUID is: ");
      const char *charUUID = pCharacteristic->getUUID().toString().c_str();
      Serial.println(charUUID);
       if(strcmp(charUUID,CHAR_TEMP_UUID) == 0){
         Serial.println("Client read Temperature!");
      }
      else if(strcmp(charUUID,CHAR_HUM_UUID) == 0){
        Serial.println("Client read Humidity!");
      }
   }
   //void onWrite(BLECharacteristic* pCharacteristic) {
    // Όταν υπάρχει χαρακτηριστικό που μπορεί να γίνει Write.
   //}
}; // MyCallbacks
//----------------------------------------------------------------

//------------------------------SETUP---------------------------------
void setup() {
  Serial.begin(115200);
  Serial.println("Starting BLE Server application...");
  BLEDevice::init(BOARD_NAME);      // Αρχικοποιώ την συσκευή BLE.
```

```
  pServer = BLEDevice::createServer();   // Δημιουργώ αντικείμενο του BLE Server.
  pServer->setCallbacks(new MyServerCallbacks());
  // Δημιουργία Service και αρχικοποίηση του με το αντίστοιχο UUID.
  BLEService* pService = pServer->createService(SERVICE_UUID);
  pCharacTemp = pService->createCharacteristic(
                                CHAR_TEMP_UUID,
                                 BLECharacteristic::PROPERTY_READ);
  pCharacTemp->setCallbacks(new MyCallbacks());        // Ορισμός της callback.

  #ifdef DHT11
  pCharacHum = pService->createCharacteristic(
                                 CHAR_HUM_UUID,
                                 BLECharacteristic::PROPERTY_READ);
  pCharacHum->setCallbacks(new MyCallbacks());

  dht.begin();                 // Ξεκινώ τον αισθητήρα DHT.
  #endif

  #ifdef DS18B20
  sensors.begin();             // Ξεκινώ τον αισθητήρα DS18B20.
  #endif
   pService->start();          // Ξεκινώ το Service.
  advertising();               // Καλείται η συνάρτηση για την διαφήμιση.
}
//----------------------------------------------------------------

//------------------------------LOOP--------------------------------
void loop() {
  if(devConnected == true){
    // DS18B20.
    #ifdef DS18B20
    sensors.requestTemperatures();               // Κάλεσμα μεθόδου requestTemperatures().
    temp = sensors.getTempCByIndex(0);           // Ανάγνωση θερμοκρασίας σε βαθμούς
Κελσίου.
```

```
// Ελέγχει εάν κάποια ανάγνωση απέτυχε.
if (isnan(temp)||(temp <= -127.0)) {              // Σε περίπτωση αδυναμίας ανάγνωσης.
  Serial.println("Failed to read from DS19B20 sensor!");
}
else{
  dtostrf(temp,1,2,tempString);        // Μετατροπή της float τιμής σε πίνακα char.
  pCharacTemp->setValue(tempString);

  Serial.print("Temperature: ");         // Εμφάνιση θερμοκρασίας στην οθόνη.
  Serial.print(tempString);
  Serial.println("°C");
  Serial.println();
}
#endif

// DHT.
#ifdef DHT11
temp = dht.readTemperature();        // Ανάγνωση θερμοκρασίας.
if(!isnan(temp)){                    // Έλεγχος αν η τιμή είναι NAN.
  dtostrf(temp,1,2,tempString);    // Μετατροπή της float τιμής σε πίνακα char.
  pCharacTemp->setValue(tempString);
  Serial.print("Temperature: ");
  Serial.print(tempString);
  Serial.println("°C");
}
else {
  Serial.println("Faild to read Temperature from DHT11 sensor!");
}
hum = dht.readHumidity();       // Ανάγνωση υγρασίας.
if(!isnan(hum)){
  dtostrf(hum,1,2,humString);       // Μετατροπή της float τιμής σε πίνακα char;
  pCharacHum->setValue(humString);
```

```
    Serial.print("Humidity: ");           // Εμφάνιση υγρασίας στην οθόνη.
    Serial.print(humString);
    Serial.println("%");
    Serial.println();                     //Αφήνω μια κενή γραμμή.
   }
   else {
    Serial.println("Faild to read Humidity from DHT11 sensor!");
   }
   #endif
 }
 delay(30000);    // Επανάληψη κάθε 30 sec.
}
//---------------------------------------------------------------------

//-----------------------------ADVERTISING-----------------------------
void advertising(){
 BLEAdvertising* pAdvertising = pServer->getAdvertising();
 pAdvertising->addServiceUUID(SERVICE_UUID);
 pAdvertising->start();      // Αρχίζω την διαφήμιση.
}

//---------------------------------------------------------------------
```

1.2 Πλακέτα που έχει τον ρόλο του BLE Client (Node 1 Gateway):

```
// Απαιτείται εγκατάσταση των ακόλουθων βιβλιοθήκων:
// - DHT Sensor Library από Adafruit: https://github.com/adafruit/DHT-sensor-library
// - Adafruit Unified Sensor από Adafruit: https://github.com/adafruit/Adafruit_Sensor
// - PubSubClient από Nick O'Leary: https://github.com/knolleary/pubsubclient

// Εισαγωγή βιβλιοθηκών.
#include <WiFi.h>           // Βιβλιοθήκη για το WiFi.
#include <PubSubClient.h>   // Βιβλιοθήκη για το MQTT.
#include <DHT.h>            // Βιβλιοθήκη για τον αισθητήρα DHT11.
#include <BLEDevice.h>      // Βιβλιοθήκη για το BLE.
```

```
#include <BLEScan.h>
#include <time.h>          // Βιβλιοθήκη για το NTP.

// DHT Sensor
#define DHTPIN 33
#define DHTTYPE DHT11
DHT dht(DHTPIN,DHTTYPE);  // Αρχικοποίηση του αισθητήρα DHT11.


// WiFi
const char* ssid = "WiFi_SSID";                 // Όνομα του wifi δικτύου.
const char* password = "WiFi_Password";   // Κωδικός για σύνδεση στο δίκτυο.


// MQTT
const char* mqtt_server = "Broker_IP_Address";    // Διεύθυνση IP του MQTT broker.
const char* clientID = "node1Gateway";


// Δήλωση μεταβλητών για να πάρω πληροφορία από τον NTP server.
const char* ntpServer = "pool.ntp.org";    // Η διεύθυνση του NTP Server.
const long gmtOffset_sec = 7200;           // Η ζώνη ώρας σε δευτερόλεπτα.
const int daylightOffset_sec = 3600;       // Η θερινή και χειμερινή αλλαγή ώρας.


// Δήλωση και αρχικοποίηση μεταβλητών.
float temperature = 0.0;         // Μεταβλητή για θερμοκρασία.
float humidity = 0.0;            // Μεταβλητή για υγρασία.
long interval = 60000;           // Κάθε πότε να στέλνω τις μετρήσεις.
unsigned long lastSend = 0;      // Τελευταία φορά που δημοσιεύτηκαν οι τιμές.
// Μεταβλητή σημαία που δείχνει αν είναι η 1η φορά που στέλνει δεδομένα το ESP32
μετά το άνοιγμά του.
bool firstBoot = true;


// Αρχικοποίηση του αντικειμένου MQTT client.
```

```
WiFiClient wifiClient;
PubSubClient mqttClient(wifiClient);


// BLE
// Δήλωση UUID και η μετρατροπήτης σε τύπο δεδομένων BLEUUID.
static BLEUUID SERVICE_UUID("96c44c0b-fe3a-44c1-8141-1b90c6b8c982");
static BLEUUID CHAR_TEMP_UUID("5578ca4c-db3e-4d14-86ac-287507abade6");
static BLEUUID CHAR_HUM_UUID("856f5b18-7c3d-4eb3-a124-c4aa72ce6892");


// Δήλωση μεταβλητών που χρειάζονται στην BLE διασύνδεση.
// Μεταβλητές σημαίες που δείχνουν αν θα προβούμε σε σύνδεση.
bool doConnect2 = false, doConnect3 = false, doConnect4 = false;
// Αναφορές στα αντικείμενα για τους server που βρέθηκαν στο scan
static BLEAdvertisedDevice *node2Device, *node3Device, *node4Device;
// Αναφορές στις συνδέσεις BLE.
static BLEClient *pClient2, *pClient3, *pClient4;
// Αναφορές στα χαρακτηριστικά των server.
static BLERemoteCharacteristic *node2RemChar, *node3RemChar, *node4RemChar1,
*node4RemChar2;
static BLEScan* pBLEScan;
// Μεταβλητές για την αποθήκευση των δεδομένων που πήραμε από τους BLE Server.
std::string node2Temp, node3Temp, node4Temp, node4Hum;


// Δήλωση μεταβλητών με τα ονόματα των topic.
const char* temp_topic1 = "home/node1/temp";   // publishing Node1
const char* hum_topic1 = "home/node1/hum";
const char* temp_topic2 = "home/node2/temp";   // BLE Node2
const char* temp_topic3 = "home/node3/temp";   // BLE Node3
const char* temp_topic4 = "home/node4/temp";   // BLE Node4
const char* hum_topic4 = "home/node4/hum";
// subscribing time. Δέχομαι τον χρόνο αποστολής από τον χρήστη με αλλαγή της
```

```
//μεταβλητής interval.
const char* time_topic = "home/time";


// Δήλωση πρόσθετων συναρτήσεων.

void wifiConnect();   // Συνάρτηση υπεύθυνη για την σύνδεση στο WiFi.
// Συνάρτηση υπεύθυνη για τον χειρισμό αποσύνδεσης από το WiFi.
void wifiDisconnected(WiFiEvent_t event, WiFiEventInfo_t info);
void mqttConnect();         // Συνάρτηση υπεύθυνη για την σύνδεση με τον MQTT broker.
// Συνάρτηση υπεύθυνη για τον χειρισμό των εισερχόμενων MQTT μηνυμάτων.
void callback(char* topic, byte* payload, unsigned int length);
void scan(int scanTime);    // Συνάρτηση υπεύθηνη για την διαδικασία του scan.
// Συνάρτηση υπεύθυνη για την δημιουργία των αντικειμένων Client.
void initBleClient();
// Συνάρτηση υπεύθυνη για την σύνδεση με τον επιθυμητό Server.
void bleConnect(BLEAdvertisedDevice* myDevice);
// Συνάρτηση υπεύθηνη για την σύνδεση με τον NTP server.
void ntpConnect();
// Συνάρτηση υπεύθηνη για τον έλεγχο ενός string αν μπορεί να μετατραπεί σε αριθμό.
int isNumber(String s);


//------------------------------CLASSES------------------------------
// Χειρισμός των BLE server που εντοπίστηκαν με την λειτουργία Scan.
class MyAdvertisedDeviceCallbacks: public BLEAdvertisedDeviceCallbacks {
   void onResult(BLEAdvertisedDevice advertisedDevice) {
     Serial.print("BLE Advertised Device found: ");
     Serial.println(advertisedDevice.toString().c_str());

     // Ελέγχουμε αν οι συσκευές είναι αυτές που ψάχνουμε.
     if
(advertisedDevice.haveServiceUUID()&&advertisedDevice.isAdvertisingService(SERVI
CE_UUID)){
       if (advertisedDevice.haveName() && (advertisedDevice.getName()=="node2") &&
```

```
(doConnect2 == false)){
        Serial.println("Found the node2.");
        node2Device = new BLEAdvertisedDevice(advertisedDevice);
        doConnect2 = true;
      }
      else if (advertisedDevice.haveName() && (advertisedDevice.getName()=="node3")
&& (doConnect3 == false)){
        Serial.println("Found the note3.");
        node3Device = new BLEAdvertisedDevice(advertisedDevice);
        doConnect3 = true;
      }
      else if (advertisedDevice.haveName() && (advertisedDevice.getName()=="node4")
&& (doConnect4 == false)){
        Serial.println("Found the note4.");
        node4Device = new BLEAdvertisedDevice(advertisedDevice);
        doConnect4 = true;
      }
      else if ((doConnect2 == true) && (doConnect3 == true) && (doConnect4 == true)){
// Σε περίπτωση που βρεθούν όλα διέκοψε το scan πρόωρα.
        pBLEScan->stop();
      }
    } // if
  } // onResult
}; // MyAdvertisedDeviceCallbacks


// Χειρισμός των καταστάσεων της σύνδεσης.
class MyClientCallback : public BLEClientCallbacks {
  void onConnect(BLEClient* pClient) {
        if(pClient->getPeerAddress()== pClient2->getPeerAddress()){
          Serial.println("Client connected with BLE Server Node2!");
  }
```

```
  else if(pClient->getPeerAddress()== pClient3->getPeerAddress()){
    Serial.println("Client connected with BLE Server Node3!");
   }
   else if(pClient->getPeerAddress()== pClient4->getPeerAddress()){
    Serial.println("Client connected with BLE Server Node4!");
   }
 }

 void onDisconnect(BLEClient* pClient) {
      if(pClient->getPeerAddress()== pClient2->getPeerAddress()){
       Serial.println("Client disconnected from BLE Server Node2!");
      doConnect2 = false;
     }
    else if(pClient->getPeerAddress()== pClient3->getPeerAddress()){
     Serial.println("Client disconnected from BLE Server Node3!");
     doConnect3 = false;
    }
    else if(pClient->getPeerAddress()== pClient4->getPeerAddress()){
     Serial.println("Client disconnected from BLE Server Node4!");
     doConnect4 = false;
    }
  }
}; // MyClientCallback
//-----------------------------------------------------------------

//------------------------------SETUP-------------------------------
void setup() {
 Serial.begin(115200);     // Ξεκινώ το serial monitor με ρυθμό baud 115200.
 Serial.println();
 dht.begin();              // Ξεκινώ τον αισθητήρα dht.


 // WiFi
 wifiConnect();            // Καλώ την συνάρτηση για σύνδεση στο wifi.
```

```
  // Καθορίζω ποια συνάρτηση καλείται σε αποσύνδεση του WiFi.
  WiFi.onEvent(wifiDisconnected, SYSTEM_EVENT_STA_DISCONNECTED);


  // MQTT
  mqttClient.setServer(mqtt_server,1883);
// Καθορίζω την συνάρτηση που θα καλείται όταν δεχόμαστε μηνύματα MQTT.
  mqttClient.setCallback (callback);

  // NTP
  ntpConnect();

  // BLE
  BLEDevice::init("node1");   // Αρχικοποιώ την συσκευή BLE και της δίνω όνομα.
  scan(5);          // Ξεκινώ το scan στέλνοντας ως όρισμα την διάρκεια σε sec.
  initBleClient();            // Δημιουργία αντικειμένων Client.
}
//-----------------------------------------------------------------

//------------------------------LOOP---------------------------------
void loop() {
  if(!mqttClient.connected()){        // Έλεγχος σύνδεσης με τον MQTT broker.
    mqttConnect();
  }
  mqttClient.loop();                  // Διαρκής έλεγχος εισερχόμενων μηνυμάτων.


  // Όταν χαθεί η BLE σύνδεση έστω και από ένα node, κάνω σύντομο scan.
  // Το κάνω στην αρχή για αν υπάρχει η δυνατότητα άμεσης λήψης μετρήσεων.
  if ((doConnect2 == false) || (doConnect3 == false) || (doConnect4 == false)){
    scan(1);
  }


   //BLE Connect
// Έλεγχος σύνδεσης με τον BLE Server.
```

```
 if (!pClient2->isConnected() && (doConnect2 == true)){
     bleConnect(node2Device);
 }
 delay(300);
 if (!pClient3->isConnected() && (doConnect3 == true)){
  bleConnect(node3Device);
 }
 delay(300);
 if (!pClient4->isConnected() && (doConnect4 == true)){
  bleConnect(node4Device);
 }
 delay(300);


 // Αποστολή μετρήσεων στον MQTT broker κάθε 1 λεπτό ή παραπάνω, ανάλογα το
interval.
 unsigned long timeNow = millis();
 if((timeNow - lastSend > interval)||(firstBoot==true)){
  lastSend = timeNow;
  firstBoot = false;

   // Διάβασμα θερμοκρασίας και μετατροπή του σε char[].
  temperature = dht.readTemperature();
  static char tempString1[8];
  if(!isnan(temperature)){              // Έλεγχος αν η τιμή είναι NAN.
   dtostrf(temperature,1,2,tempString1);  // Μετατροπή της float τιμής σε πίνακα char.

   // Εμφάνιση θερμοκρασίας του Node 1.
   Serial.print("Temperature node 1: ");
   Serial.print(tempString1);
   Serial.println("°C");

   // Στέλνω ως ορίσματα το όνομα του topic και τον πίνακα char με την τιμή.
   mqttClient.publish(temp_topic1, tempString1);
```

```
  }
  else {
    Serial.println("Faild to read Temperature from DHT11 sensor!");
  }

  // Διάβασμα υγρασίας και μετατροπή του σε char[].
  humidity = dht.readHumidity();
  static char humString1[8];
  if(!isnan(humidity)){
    dtostrf(humidity,1,2,humString1);
    // Εμφάνιση υγρασίας του Node 1.
    Serial.print("Humidity node 1: ");
    Serial.print(humString1);
    Serial.println("%");
    Serial.println();
    mqttClient.publish(hum_topic1, humString1);
  }
  else {
    Serial.println("Faild to read Humidity from DHT11 sensor!");
  }

  static char tempString2[8];
  if (pClient2->isConnected()){
    node2Temp = node2RemChar->readValue();
    // Για να αποφυγή αποστολής 0 σε περίπτωση που λάβω κενό μήνυμα. Όταν λάβω
//κενό μήνυμα δεν στέλνω τίποτα.
     if((node2Temp != "")){
       strcpy(tempString2, node2Temp.c_str());
       // Εμφάνιση μετρήσεων από το Node 2.
      Serial.print("Temperature node 2: ");
```

```
    Serial.print(tempString2);
    Serial.println("°C");
    Serial.println("");   //Αφήνω μια κενή γραμμή.
    mqttClient.publish(temp_topic2, tempString2);    // BLE
    }
}

static char tempString3[8];
if (pClient3->isConnected()){
  node3Temp = node3RemChar->readValue();
  if((node3Temp != "")){
    strcpy(tempString3, node3Temp.c_str());
   // Εμφάνιση μετρήσεων από το Node 3.
    Serial.print("Temperature node 3: ");
    Serial.print(tempString3);
    Serial.println("°C");
    Serial.println("");
    mqttClient.publish(temp_topic3, tempString3);    // BLE
    }
}

static char tempString4[8];
static char humString4[8];
if (pClient4->isConnected()){
  node4Temp = node4RemChar1->readValue();
  node4Hum = node4RemChar2->readValue();
  if((node4Temp != "") && (node4Hum != "")){
    strcpy(tempString4, node4Temp.c_str());
    strcpy(humString4, node4Hum.c_str());
```

```
      // Εμφάνιση μετρήσεων από το Node 4.
      Serial.print("Temperature node 4: ");
      Serial.print(tempString4);
      Serial.println("°C");
      Serial.print("Humidity node 4: ");
      Serial.print(humString4);
      Serial.println("%");
      Serial.println();

      mqttClient.publish(temp_topic4, tempString4);    // BLE
      mqttClient.publish(hum_topic4, humString4);
      }
    }
  }
}
//---------------------------------------------------------------------

//-----------------------------WIFICONNECT----------------------------
void wifiConnect(){
  // Αποσύνδεση από παλιό δίκτυο σε περίπτωση που υπήρχε σύνδεση wifi.
  WiFi.disconnect(true);
  WiFi.mode(WIFI_STA);              // Ορίζω την πλακέτα ως access point.
  Serial.print("Connecting to WiFi: ");

  Serial.print(ssid);
  WiFi.begin(ssid,password);          // Αρχίζω την σύνδεση με το δίκτυο WiFi.


// Λήψη της κατάστασης της σύνδεσης WiFi και έλεγχος για το πότε θα γίνει η σύνδεση.
  while(WiFi.status() != WL_CONNECTED){
    delay(500);
    Serial.print(".");
  }
  Serial.println();
  Serial.print("Node1 IP Address: ");
```

```
  Serial.println(WiFi.localIP());         // Εκτύπωση της IP διεύθυνσης.
}
//-------------------------------------------------------------------

//-----------------------------WIFIDISCONNECT--------------------------
void wifiDisconnected(WiFiEvent_t event, WiFiEventInfo_t info){
  Serial.println("Disconnected from WiFi");
  Serial.print("Reason: ");
  // Κωδικοποιημένος ο λόγος αποσύνδεσης από το WiFi.
  Serial.println(info.disconnected.reason);
  Serial.println("Attempt to reconnect wifi");
}
//-------------------------------------------------------------------

//-----------------------------MQTTCONNECT-----------------------------
void mqttConnect(){
  // Επαναλαμβάνει μέχρι να συνδεθεί στον MQTT broker.
  while(!mqttClient.connected()){
    Serial.println("Attempting MQTT connection...");
   // Καλώ την συνάρτηση σύνδεσης στέλνοντας το όνομα του client και μου επιστρέφει
//true ή false.
    if(mqttClient.connect(clientID)){
      Serial.println("Connected to MQTT Broker");


      // Εγγραφή στα topic από τα οποία θα λαμβάνω μηνύματα.
      mqttClient.subscribe(time_topic);
    }
    else{
      Serial.print("Failed, rc=");
      Serial.print(mqttClient.state());
      Serial.println(" try again in 5 seconds");
     // Περιμένει 5 δευτερόλεπτα πριν ξαναπροσπαθήσει για σύνδεση στον MQTT Broker.
```

```
      delay(5000);
    }
  }
}
//--------------------------------------------------------------------

//-----------------------------CALLBACK---------------------------------
void callback(char* topic, byte* payload, unsigned int length){
  Serial.print("Message arrived on topic: ");
  Serial.println(topic);


  String message;
  for(int i=0;i<length;i++){                // Διατρέχω ένα ένα τα στοιχεία του payload.
    // Μετατρέπω τα byte σε char και τα συντάσω σε ένα ενιαιο string.
    message = message + (char)payload[i];
  }
  Serial.print("Message: ");
  Serial.println(message);                  // Εκτύπωση μηνύματος που έλαβα.


  if((String)topic == time_topic){          // Αν το μήνυμα είναι για το topic του time.
    if(isNumber(message) == 1){             // Ελέγχει αν είναι αριθμός.
      // Μετατροπή της συμβολοσειράς που περιέχει το μήνυμα σε intenger.
      int timer = atoi(message.c_str());
  // Αν η τιμή που λάβαμε είναι μεγαλύτερη του 1 τότε προχωράμε σε αλλαγή του interval.
      if(timer >= 1){
// Επειδή λαμβάουμε την τιμή σε λεπτά την μετατρέπω σε millisecond και την εκχωρώ
//στο interval.
        interval = 60*1000*long(timer);
        Serial.print("Changing data transmission time to ");
        Serial.print(interval);
        Serial.println(" millisecond.");
      }
```

```
    }
  }
}
//------------------------------------------------------------------------

//------------------------------SCAN-----------------------------------
void scan(int scanTime){
  //Δημιουργία αντικειμένου της κλάσης BLEScan.
  pBLEScan = BLEDevice::getScan();
  // Το αρχικοποιώ με την συνάρτηση που θα καλείτε όταν ανιχνεύει κάποιον server.
  pBLEScan->setAdvertisedDeviceCallbacks(new MyAdvertisedDeviceCallbacks());
  // Επιλογή μιας ενεργής σάρωσης για πιο γρήγορα απότελέσματα.
  pBLEScan->setActiveScan(true);
  // Ορίζω το διάστημα μεταξύ των σαρώσεων σε millisecond.
  pBLEScan->setInterval(1349);
  // Ρυθμίζω το παράθυρο για ενεργή σάρωση σε millisecond.
  pBLEScan->setWindow(449);
  BLEScanResults foundDevices = pBLEScan->start(scanTime, false);
}
//------------------------------------------------------------------------

//------------------------------INITBLECLIENT-----------------------------
void initBleClient(){       // Δημιουργώ τα αντικείμενα client και τους ορίζω την callback.
    pClient2  = BLEDevice::createClient();
    Serial.println(" - Created client2");
   // Καθορίζω ποια συνάρτηση καλείται σε αποσύνδεση από τον BLE server.
    pClient2->setClientCallbacks(new MyClientCallback());
    delay(50);
    pClient3  = BLEDevice::createClient();
    Serial.println(" - Created client3");
    pClient3->setClientCallbacks(new MyClientCallback());
    delay(50);
    pClient4  = BLEDevice::createClient();
    Serial.println(" - Created client4");
```

```
  pClient4->setClientCallbacks(new MyClientCallback());
}
//----------------------------------------------------------------------

//------------------------------BLECONNECT------------------------------
// Συνάρτηση που είναι υπεύθυνη για την σύνδεση με τους server.
void bleConnect(BLEAdvertisedDevice* myDevice){
 if (myDevice->getName()=="node2"){                // Ελέγχει το όνομα της συσκευής.
   Serial.print("Forming a connection to ");
  // Εκτυπώνει την μοναδική BLE διεύθυνσή της.
   Serial.println(myDevice->getAddress().toString().c_str());
   pClient2->connect(myDevice);    // Δημιουργία BLE σύνδεσης με τον server.
   Serial.println(" - Connected to server Node2");
   // Δημιουργία αναφοράς στο Service του Server.
   BLERemoteService* pRemoteService2 = pClient2->getService(SERVICE_UUID);
   Serial.println(" - Connected to service of Node2");
   // Δημιουργία αναφοράς στο χαρακτηριστικό της θερμοκρασίας.
   node2RemChar = pRemoteService2->getCharacteristic(CHAR_TEMP_UUID);
   Serial.println(" - Connected to Temperature characteristic of Node2");
   return;
 }
 else if(myDevice->getName()=="node3"){
   Serial.print("Forming a connection to ");
   Serial.println(myDevice->getAddress().toString().c_str());
   pClient3->connect(myDevice);
   Serial.println(" - Connected to server Node3");
   BLERemoteService* pRemoteService3 = pClient3->getService(SERVICE_UUID);
   Serial.println(" - Connected to service of Node3");
   node3RemChar = pRemoteService3->getCharacteristic(CHAR_TEMP_UUID);
   Serial.println(" - Connected to Temperature characteristic of Node3");
   return;
 }
 else if(myDevice->getName()=="node4"){
```

```
    Serial.print("Forming a connection to ");
    Serial.println(myDevice->getAddress().toString().c_str());
    pClient4->connect(myDevice);
    Serial.println(" - Connected to server Node4");
    BLERemoteService* pRemoteService4 = pClient4->getService(SERVICE_UUID);
    Serial.println(" - Connected to service of Node4");
    node4RemChar1 = pRemoteService4->getCharacteristic(CHAR_TEMP_UUID);
    Serial.println(" - Connected to Temperature characteristic of Node4");
    delay(100);
    // Δημιουργία αναφοράς στο χαρακτηριστικό της υγρασίας.
    node4RemChar2 = pRemoteService4->getCharacteristic(CHAR_HUM_UUID);
    Serial.println(" - Connected to Humidity characteristic of Node4");
    return;
  }
}
//----------------------------------------------------------------------

//-----------------------------ntpCONNECT-------------------------------
  void ntpConnect(){
    // Αρχικοποίηση των στοιχείων για σύνδεση με τον NTP server και λήψη της χρονικής
//σήμανσης.
    configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
    // Δημιουργία δομής τύπου tm, για χειρισμό των δεδομένων που ελήφθησαν.
    struct tm timeInfo;
    // Λαμβάνω την ημερομηνία από τον RTC και αρχικοποιώ την timeInfo.
    if(!getLocalTime(&timeInfo)){
      Serial.println("Failed to obtain time");
      return;
    }
    // Με το H κεφαλαίο επιστρέφεται η ώρα στο 24ο format.
    Serial.println(&timeInfo, "%A, %d %B %Y %H:%M:%S");          }

//----------------------------------------------------------------------
```

```
//------------------------------isNumber---------------------------------
// Σε περίπτωση που έστω ένα στοιχείο δεν είναι αριθμός επιστρέφεται 0, αν όλα είναι
//αριθμοί επιστρέφεται 1.
int isNumber(String s){
  for(int i = 0;i<strlen(s.c_str());i++){  // Ελέγχει ένα ένα τα στοιχεία του string.
  // Με το isdigit() εάν ένας χαρακτήρας είναι αριθμητικός χαρακτήρας (0-9) επιστρέφει 1,
//αλλίως επιστρέφει 0.
    if(isdigit(s[i]) == 0){
      return 0;
    }
  }
  return 1;                          // Όλα τα ψηφία είναι αριθμητικοί χαρακτήρες από (0-9).
}
//---------------------------------------------------------------------
```

2 Υποσύστημα Εξωτερικού Χώρου (Node 5)

```
// Απαιτείται εγκατάσταση των ακόλουθων βιβλιοθήκων:
// - DHT Sensor Library από Adafruit: https://github.com/adafruit/DHT-sensor-library
// - Adafruit Unified Sensor από Adafruit: https://github.com/adafruit/Adafruit_Sensor
// - PubSubClient από Nick O'Leary: https://github.com/knolleary/pubsubclient
// - BH1750 από Christopher Laws: https://github.com/claws/BH1750

// Εισαγωγή βιβλιοθηκών.
#include <WiFi.h>          // Βιβλιοθήκη για το WiFi.
#include <PubSubClient.h>  // Βιβλιοθήκη για το MQTT.
#include <DHT.h>           // Βιβλιοθήκη για τον αισθητήρα DHT11.
#include <time.h>          // Βιβλιοθήκη για το NTP.
#include <Wire.h>          // Βιβλιοθήκη για επικοινωνία με συσκευές I2C.
#include <BH1750.h>        // Βιβλιοθήκη για τον αισθητήρα BH1750.

// DHT Sensor
```

```
#define DHTPIN 33                  // Pin για τον DHT11.
#define DHTTYPE DHT11
DHT dht(DHTPIN,DHTTYPE);  // Αρχικοποίηση του αισθητήρα DHT11.


// BH1750 Sensor
BH1750 lightMeter;        // Αρχικοποίηση του αισθητήρα BH1750.


// WiFi
const char* ssid = "WiFi_SSID";                 // Όνομα του wifi δικτύου.
const char* password = "WiFi_Password";   // Κωδικός για σύνδεση στο δίκτυο.


// MQTT
const char* mqtt_server = "Broker_IP_Address";    // Διεύθυνση IP του MQTT broker.
const char* clientID = "node5";        // Όνομα που δίνουμε στην πλακέτα ως MQTT client.


// Δήλωση μεταβλητών για να πάρω πληροφορία από τον NTP server.
const char* ntpServer = "pool.ntp.org";    // Η διεύθυνση του NTP Server.
const long gmtOffset_sec = 7200;           // Η ζώνη ώρας σε δευτερόλεπτα.
const int daylightOffset_sec = 3600;       // Η θερινή και χειμερινή αλλαγή ώρας.


// Δήλωση και αρχικοποίηση μεταβλητών.
float temperature = 0.0;         // Μεταβλητή για θερμοκρασία.
float humidity = 0.0;            // Μεταβλητή για υγρασία.
float irradiation = 0.0;         // Μεταβλητή για ηλιακή ακτινοβολία.
long interval = 60000;           // Κάθε πότε να στέλνω τις μετρήσεις.
unsigned long lastSend = 0;      // Τελευταία φορά που δημοσιεύτηκαν οι τιμές.
// Μεταβλητή σημαία που δείχνει αν είναι η 1η φορά που στέλνει δεδομένα το ESP32
//μετά το άνοιγμά του.
bool firstBoot = true;


// Αρχικοποίηση του αντικειμένου MQTT client.
```

```
WiFiClient wifiClient;
PubSubClient mqttClient(wifiClient);


// Δήλωση μεταβλητών με τα ονόματα των topic.
const char* temp_topic = "home/node5/temp";      // publishing Node5
const char* hum_topic = "home/node5/hum";
const char* sun_topic = "home/node5/sun";
// subscribing time. Δέχομαι τον χρόνο αποστολής από τον χρήστη με αλλαγή της
//μεταβλητής interval.
const char* time_topic = "home/time";


// Δήλωση πρόσθετων συναρτήσεων.
void wifiConnect();         // Συνάρτηση υπεύθυνη για την σύνδεση στο WiFi.
// Συνάρτηση υπεύθυνη για τον χειρισμό αποσύνδεσης από το WiFi.
void wifiDisconnected(WiFiEvent_t event, WiFiEventInfo_t info);
void mqttConnect();         // Συνάρτηση υπεύθυνη για την σύνδεση με τον MQTT broker.
// Συνάρτηση υπεύθυνη για τον χειρισμό των εισερχόμενων MQTT μηνυμάτων.
void callback(char* topic, byte* payload, unsigned int length);
// Συνάρτηση υπεύθηνη για την σύνδεση με τον NTP server και εκτύπωση της τρέχουσας
//ώρας.
void ntpConnect();
// Συνάρτηση υπεύθηνη για τον έλεγχο ενός string αν μπορεί να μετατραπεί σε αριθμό.
int isNumber(String s);


//------------------------------SETUP------------------------------
void setup() {
  Serial.begin(115200);     // Ξεκινώ το serial monitor με ρυθμό baud 115200.
  Serial.println();
  dht.begin();              // Ξεκινώ τον αισθητήρα dht.
```

```
  // Αρχικοποίηση του διαύλου I2C (η βιβλιοθήκη BH1750 δεν το κάνει αυτόματα)
  Wire.begin();
   lightMeter.begin();       // Ξεκινώ τον αισθητήρα BH1750 διαμέσου του lightMeter.


   // WiFi
  wifiConnect();            // Καλώ την συνάρτηση για σύνδεση στο wifi.
  // Καθορίζω ποια συνάρτηση καλείται σε αποσύνδεση του WiFi.
  WiFi.onEvent(wifiDisconnected, SYSTEM_EVENT_STA_DISCONNECTED);


  // MQTT
  // Αρχικοποιώ το αντικείμενο MQTT client με πληροφορίες για τον server.
  mqttClient.setServer(mqtt_server,1883);
  // Καθορίζω την συνάρτηση που θα καλείται όταν δεχόμαστε μηνύματα από τον MQTT
//broker.
  mqttClient.setCallback (callback);


  // NTP
  ntpConnect();             // Σύνδεση και λήψη ώρας από τον NTP Server.
}
//--------------------------------------------------------------------

//-----------------------------LOOP--------------------------------
void loop() {
  if(!mqttClient.connected()){        // Έλεγχος σύνδεσης με τον MQTT broker.
    mqttConnect();
  }
  mqttClient.loop();                  // Διαρκής έλεγχος εισερχόμενων μηνυμάτων.


  // Αποστολή μετρήσεων στον MQTT broker κάθε 1 λεπτό ή παραπάνω, ανάλογα το
//interval.
  unsigned long timeNow = millis();
  if((timeNow - lastSend > interval)||(firstBoot==true)){
```

```
  lastSend = timeNow;
  firstBoot = false;

  // Διάβασμα θερμοκρασίας και μετατροπή του σε char[].
  temperature = dht.readTemperature();
  static char tempString[8];
  if(!isnan(temperature)){              // Έλεγχος αν η τιμή είναι NAN.
    dtostrf(temperature,1,2,tempString);   // Μετατροπή της float τιμής σε πίνακα char.

    // Εμφάνιση Θερμοκρασίας του Node 5.
    Serial.print("Temperature node 5: ");
    Serial.print(tempString);
    Serial.println("°C");
    // Στέλνω ως ορίσματα το όνομα του topic που δημιουργώ και τον πίνακα char με την
//τιμή.
    mqttClient.publish(temp_topic, tempString);
  }
  else {
    Serial.println("Faild to read Temperature from DHT11 sensor!");
  }

  // Διάβασμα υγρασίας και μετατροπή του σε char[].
  humidity = dht.readHumidity();
  static char humString[8];
  if(!isnan(humidity)){
    dtostrf(humidity,1,2,humString);

    // Εμφάνιση υγρασίας του Node 5.
    Serial.print("Humidity node 5: ");
    Serial.print(humString);
```

```
    Serial.println("%");
    Serial.println();

    mqttClient.publish(hum_topic, humString);
   }
   else {
    Serial.println("Faild to read Humidity from DHT11 sensor!");
   }


   // Διάβασμα φωτεινότητας και η ματατροπή της σε ηλιακή ακτινοβολία.
   float lux = lightMeter.readLightLevel();
   static char sunString[8];
  // Σε περίπτωση αδυναμίας ανάγνωσης ο αισθητήρας BH1750 σε πολλές περιπτώσεις
//επιστρέφει -2.00 ή -1.00.
   if (isnan(lux)||(lux <= -1.00)) {

    Serial.println("Failed to read from BH1750 sensor!");
   }
   else{
    Serial.print("lux illuminance: ");
    Serial.print(lux);
    Serial.println("lx");


     // Υποθέτω ότι ο συντελεστής μετατροπής ακτινοβολίας σε φωτεινότητα είναι 1
//W/m2 ίση με 122±1 lx για εξωτερικό φυσικό ηλιακό φως.
     irradiation = lux*0.0082;
    dtostrf(irradiation,1,2,sunString);   // Μετατροπή της float τιμής σε πίνακα char.

    Serial.print("Solar Irradiance: ");
    Serial.print(sunString);
    Serial.println("W/m2");
    Serial.println();

    mqttClient.publish(sun_topic, sunString);
   }
```

```
  }
}
//-------------------------------------------------------------------------

//----------------------------WIFICONNECT----------------------------
void wifiConnect(){
  // Αποσύνδεση από παλιό δίκτυο σε περίπτωση που υπήρχε σύνδεση wifi.
  WiFi.disconnect(true);
  WiFi.mode(WIFI_STA);                // Ορίζω την πλακέτα ως access point και station.
  Serial.print("Connecting to WiFi: ");
  Serial.print(ssid);
  WiFi.begin(ssid,password);           // Αρχίζω την σύνδεση με το δίκτυο WiFi.

   // Λήψη της κατάστασης της σύνδεσης WiFi και έλεγχος για το πότε θα γίνει η σύνδεση.
  while(WiFi.status() != WL_CONNECTED){
    delay(500);
    Serial.print(".");
  }
  Serial.println();
  Serial.print("Node5 IP Address: ");
  Serial.println(WiFi.localIP());        // Εκτύπωση της IP διεύθυνσης.
}
//-------------------------------------------------------------------------

//----------------------------WIFIDISCONNECT----------------------------
// Εκτελείται παράλληλα με την συνάρτηση αποσύνδεσης MQTT.
void wifiDisconnected(WiFiEvent_t event, WiFiEventInfo_t info){
  Serial.println("Disconnected from WiFi");
  Serial.print("Reason: ");
  // Κωδικοποιημένος ο λόγος αποσύνδεσης από το WiFi.
  Serial.println(info.disconnected.reason);
  Serial.println("Attempt to reconnect wifi");
}
//-------------------------------------------------------------------------
```

```
//------------------------------MQTTCONNECT-------------------------------
void mqttConnect(){
 // Επαναλαμβάνει μέχρι να συνδεθεί στον MQTT broker.
 while(!mqttClient.connected()){
  Serial.println("Attempting MQTT connection...");
  // Καλώ την συνάρτηση σύνδεσης στέλνοντας το όνομα του client και μου επιστρέφει
//true ή false.
  if(mqttClient.connect(clientID)){
   Serial.println("Connected to MQTT Broker");

   // Εγγραφή στα topic από τα οποία θα λαμβάνω μηνύματα.
   mqttClient.subscribe(time_topic);
  }
  else{       // Σε αποτυχία σύνδεσης με τον MQTT εκτυπώνω των κωδικό σφάλαμτος.
   Serial.print("Failed, rc=");
   // Ελέγχω την κατάσταση του client όσο αφορά την σύνδεση με τον MQTT.
   Serial.print(mqttClient.state());
   Serial.println(" try again in 5 seconds");
   delay(5000);
  }
 }
}
//-----------------------------------------------------------------------

//------------------------------CALLBACK---------------------------------
void callback(char* topic, byte* payload, unsigned int length){
 Serial.print("Message arrived on topic: ");
 Serial.println(topic);

 String message;
 for(int i=0;i<length;i++){              // Διατρέχω ένα ένα τα στοιχεία του payload.
  // Μετατρέπω τα byte σε char και τα συντάσω σε ένα ενιαιο string.
  message = message + (char)payload[i];
 }
```

```
  Serial.print("Message: ");
  Serial.println(message);                  // Εκτύπωση μηνύματος που έλαβα.


  if((String)topic == time_topic){          // Αν το μήνυμα είναι για το topic του time.
    if(isNumber(message) == 1){             // Ελέγχει αν είναι αριθμός.
      // Μετατροπή της συμβολοσειράς που περιέχει το μήνυμα σε intenger.
      int timer = atoi(message.c_str());
      if(timer >= 1){
        interval = 60*1000*long(timer);
        Serial.print("Changing data transmission time to ");
        Serial.print(interval);
        Serial.println(" millisecond.");
      }
    }
  }
}
//---------------------------------------------------------------------

//-----------------------------ntpCONNECT-------------------------------
  void ntpConnect(){
    // Αρχικοποίηση των στοιχείων για σύνδεση με τον NTP server και λήψη της χρονικής
//σήμανσης.
    configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
    struct tm timeInfo;     // Δημιουργία δομής τύπου tm.
    // Λαμβάνω την ημερομηνία από τον RTC και αρχικοποιώ την timeInfo.
    if(!getLocalTime(&timeInfo)){
      Serial.println("Failed to obtain time");
      return;
    }
    Serial.println(&timeInfo, "%A, %d %B %Y %H:%M:%S");
  }
//---------------------------------------------------------------------
```

```
//------------------------------isNumber---------------------------------
// Σε περίπτωση που έστω ένα στοιχείο δεν είναι αριθμός επιστρέφεται 0, αν όλα είναι
//αριθμοί επιστρέφεται 1.
int isNumber(String s){
  for(int i = 0;i<strlen(s.c_str());i++){  // Ελέγχει ένα ένα τα στοιχεία του string.
    // Με το isdigit() εάν ένας χαρακτήρας είναι αριθμητικός χαρακτήρας (0-9) επιστρέφει
//1, αλλίως επιστρέφει 0.
    if(isdigit(s[i]) == 0){
      return 0;
    }
  }
  return 1;                          // Όλα τα ψηφία είναι αριθμητικοί χαρακτήρες από (0-9).
}
//---------------------------------------------------------------------
```

3 Υποσύστημα Λεβητοστασίου (Node 6)

```
// Απαιτείται εγκατάσταση των ακόλουθων βιβλιοθήκων:
// - DHT Sensor Library από Adafruit: https://github.com/adafruit/DHT-sensor-library
// - Adafruit Unified Sensor από Adafruit: https://github.com/adafruit/Adafruit_Sensor
// - PubSubClient από Nick O'Leary: https://github.com/knolleary/pubsubclient
// - OneWire από Paul Stoffregen: https://github.com/PaulStoffregen/OneWire
// - DallasTemperature από Miles Burton: https://github.com/milesburton/Arduino-
Temperature-Control-Library


// Εισαγωγή βιβλιοθηκών.
#include <WiFi.h>          // Βιβλιοθήκη για το WiFi.
#include <PubSubClient.h>  // Βιβλιοθήκη για το MQTT.
#include <DHT.h>           // Βιβλιοθήκη για τον αισθητήρα DHT11.
#include <OneWire.h>       // Βιβλιοθήκες για τον αισθητήρα DS18B20.
#include <DallasTemperature.h>
#include <time.h>          // Βιβλιοθήκη για το NTP.
```

```
// DHT Sensor
#define DHTPIN 33         // Ορίζω το pin του αισθητήρα DHT11.
#define DHTTYPE DHT11
DHT dht(DHTPIN,DHTTYPE);  // Αρχικοποίηση του αισθητήρα DHT11.

// DS18B20 Sensor
const int oneWireBus = 18;       // Ορίζω το pin του αισθητήρα DS18B20.
OneWire oneWire(oneWireBus);
DallasTemperature sensors(&oneWire);


// Relay
#define RELAYPIN 19    // Ορίζω το pin για το relay.


// WiFi
const char* ssid = "WiFi_SSID";                 // Όνομα του wifi δικτύου.
const char* password = "WiFi_Password";   // Κωδικός για σύνδεση στο δίκτυο.


// MQTT
const char* mqtt_server = "Broker_IP_Address";    // Διεύθυνση IP του MQTT broker.
const char* clientID = "node6";              // Όνομα που δίνουμε στην πλακέτα.


// Δήλωση μεταβλητών για να πάρω πληροφορία από τον NTP server.
const char* ntpServer = "pool.ntp.org";    // Η διεύθυνση του NTP Server.
const long gmtOffset_sec = 7200;           // Η ζώνη ώρας σε δευτερόλεπτα.
const int daylightOffset_sec = 3600;       // Η θερινή και χειμερινή αλλαγή ώρας.


// Δήλωση και αρχικοποίηση μεταβλητών.
float tempDHT11 = 0.0;         // Μεταβλητή για θερμοκρασία του DHT11.
float humDHT11 = 0.0;          // Μεταβλητή για υγρασία του DHT11.
float tempDS18B20 = 0.0;       // Μεταβλητή για θερμοκρασία του DS18B20.
long interval = 60000;         // Κάθε πότε να στέλνω τις μετρήσεις.
```

```
unsigned long lastSend = 0;    // Τελευταία φορά που δημοσιεύτηκαν οι τιμές.
// Μεταβλητή σημαία που δείχνει αν είναι η 1η φορά που στέλνει δεδομένα το ESP32
//μετά το άνοιγμά του.
bool firstBoot = true;


// Αρχικοποίηση του αντικειμένου MQTT client.
WiFiClient wifiClient;
PubSubClient mqttClient(wifiClient);


// Δήλωση μεταβλητών με τα ονόματα των topic.
const char* temp_topic1 = "home/node6/tempDHT";    // publishing Node6
const char* hum_topic1 = "home/node6/humDHT";
const char* temp_topic2 = "home/node6/tempDS";
const char* relay_topic = "home/node6/relay";      // subscribing Node6 .
// subscribing time. Δέχομαι τον χρόνο αποστολής από τον χρήστη με αλλαγή της
//μεταβλητής interval.
const char* time_topic = "home/time";


// Δήλωση πρόσθετων συναρτήσεων.
void wifiConnect();         // Συνάρτηση υπεύθυνη για την σύνδεση στο WiFi.
// Συνάρτηση υπεύθυνη για τον χειρισμό αποσύνδεσης από το WiFi.
void wifiDisconnected(WiFiEvent_t event, WiFiEventInfo_t info);
void mqttConnect();         // Συνάρτηση υπεύθυνη για την σύνδεση με τον MQTT broker.
// Συνάρτηση υπεύθυνη για τον χειρισμό των εισερχόμενων MQTT μηνυμάτων.
void callback(char* topic, byte* payload, unsigned int length);
// Συνάρτηση υπεύθηνη για την σύνδεση με τον NTP server και εκτύπωση της τρέχουσας
//ώρας.
void ntpConnect();
int isNumber(String s);     // Συνάρτηση υπεύθηνη για τον έλεγχο ενός string αν μπορεί να
//μετατραπεί σε αριθμό.
```

```
//------------------------------SETUP------------------------------
void setup() {
  Serial.begin(115200);     // Ξεκινώ το serial monitor με ρυθμό baud 115200.
  Serial.println();
  dht.begin();              // Ξεκινώ τον αισθητήρα dht.
  sensors.begin();          // Ξεκινώ τον αισθητήρα DS18B20.

  // Relay
  pinMode(RELAYPIN,OUTPUT);     // Δηλώνω το pin που θα είναι ως έξοδος.
  digitalWrite(RELAYPIN,HIGH);  // Το αρχικοποιώ στην τιμή στο "OFF".


  // WiFi
  wifiConnect();            // Καλώ την συνάρτηση για σύνδεση στο wifi.
  // Καθορίζω ποια συνάρτηση καλείται σε αποσύνδεση του WiFi.
  WiFi.onEvent(wifiDisconnected, SYSTEM_EVENT_STA_DISCONNECTED);


  // MQTT
  mqttClient.setServer(mqtt_server,1883);
  // Καθορίζω την συνάρτηση που θα καλείται όταν δεχόμαστε μηνύματα από τον MQTT
//broker.
  mqttClient.setCallback (callback);

  // NTP
  ntpConnect();
}
//-----------------------------------------------------------------

//------------------------------LOOP------------------------------
void loop() {
  if(!mqttClient.connected()){        // Έλεγχος σύνδεσης με τον MQTT broker.
    mqttConnect();
  }
  mqttClient.loop();                  // Διαρκής έλεγχος εισερχόμενων μηνυμάτων.
```

```
 // Αποστολή μετρήσεων στον MQTT broker κάθε 1 λεπτό ή παραπάνω, ανάλογα το
//interval.
 unsigned long timeNow = millis();
 if((timeNow - lastSend > interval)||(firstBoot==true)){
  lastSend = timeNow;
  firstBoot = false;


  // DHT11.
  tempDHT11 = dht.readTemperature();
  static char tempString1[8];
  if(!isnan(tempDHT11)){              // Έλεγχος αν η τιμή είναι NAN.
   dtostrf(tempDHT11,1,2,tempString1);  // Μετατροπή της float τιμής σε πίνακα char.
   Serial.print("Temperature DHT11: ");
   Serial.print(tempString1);
   Serial.println("°C");
  // Στέλνω ως ορίσματα το όνομα του topic και τον πίνακα char με την τιμή.
  mqttClient.publish(temp_topic1, tempString1);
  }
  else {
   Serial.println("Faild to read Temperature from DHT11 sensor!");
  }


  // Διάβασμα υγρασίας και μετατροπή του σε char[].
  humDHT11 = dht.readHumidity();
  static char humString1[8];
  if(!isnan(humDHT11)){
   dtostrf(humDHT11,1,2,humString1);
   Serial.print("Humidity DHT11: ");
   Serial.print(humString1);
   Serial.println("%");
   Serial.println();
```

```
    mqttClient.publish(hum_topic1, humString1);
   }
   else {
    Serial.println("Faild to read Humidity from DHT11 sensor!");
   }


  // DS18B20.
  sensors.requestTemperatures();              // Κλήση μεθόδου requestTemperatures().
  tempDS18B20 = sensors.getTempCByIndex(0);
  static char tempString2[8];
  // Ελέγχει εάν κάποια ανάγνωση απέτυχε.
  if (isnan(tempDS18B20)||(tempDS18B20 <= -127.0)) {
     Serial.println("Failed to read from DS19B20 sensor!");
   }
   else{
     dtostrf(tempDS18B20,1,2,tempString2);
     Serial.print("Temperature DS18B20: ");
     Serial.print(tempString2);
     Serial.println("°C");
     Serial.println();
     mqttClient.publish(temp_topic2, tempString2);
   }
  }
}
//-------------------------------------------------------------------------------
//------------------------------WIFICONNECT-----------------------------
void wifiConnect(){
  // Αποσύνδεση από παλιό δίκτυο σε περίπτωση που υπήρχε σύνδεση wifi.
  WiFi.disconnect(true);
  WiFi.mode(WIFI_STA);                // Ορίζω την πλακέτα ως access point.
```

```
  Serial.print("Connecting to WiFi: ");
  Serial.print(ssid);
  WiFi.begin(ssid,password);              // Αρχίζω την σύνδεση με το δίκτυο WiFi.


  // Λήψη της κατάστασης της σύνδεσης WiFi και έλεγχος για το πότε θα γίνει η σύνδεση.
  while(WiFi.status() != WL_CONNECTED){
    delay(500);
    Serial.print(".");
  }
  Serial.println();
  Serial.print("Node6 IP Address: ");
  Serial.println(WiFi.localIP());
}
//-------------------------------------------------------------------------

//-----------------------------WIFIDISCONNECT---------------------------
void wifiDisconnected(WiFiEvent_t event, WiFiEventInfo_t info){
  Serial.println("Disconnected from WiFi");
  Serial.print("Reason: ");
  Serial.println(info.disconnected.reason);
  Serial.println("Attempt to reconnect wifi");
}
//-------------------------------------------------------------------------

//-----------------------------MQTTCONNECT------------------------------

void mqttConnect(){
  // Επαναλαμβάνει μέχρι να συνδεθεί στον MQTT broker.
  while(!mqttClient.connected()){
    Serial.println("Attempting MQTT connection...");
    if(mqttClient.connect(clientID)){
      Serial.println("Connected to MQTT Broker");

      // Εγγραφή στα topic από τα οποία θα λαμβάνω μηνύματα.
      mqttClient.subscribe(relay_topic);
```

```
    mqttClient.subscribe(time_topic);
   }
   else{
    Serial.print("Failed, rc=");
    Serial.print(mqttClient.state());
    Serial.println(" try again in 5 seconds");
    delay(5000);
   }
  }
}
//--------------------------------------------------------------------------------

//-----------------------------CALLBACK---------------------------------
void callback(char* topic, byte* payload, unsigned int length){
  Serial.print("Message arrived on topic: ");
  Serial.println(topic);


  String message;
  for(int i=0;i<length;i++){              // Διατρέχω ένα ένα τα στοιχεία του payload.
   message = message + (char)payload[i];
  }
  Serial.print("Message: ");
  Serial.println(message);                // Εκτύπωση μηνύματος που έλαβα.


  // Όταν έρθει μήνυμα που αφορά το relay.
  if((String)topic == relay_topic){       // Αν το μήνυμα είναι για το topic του relay.
   Serial.print("Changing relay to ");
   if(message == "ON"){                   // Αν γράφει ON, άναψε.
    digitalWrite(RELAYPIN,LOW);           // τα relay έχουν active στο LOW.
    Serial.println("ON");
   }
   else if(message == "OFF"){             // Αν γράφει OFF, σβήσε.
    digitalWrite(RELAYPIN,HIGH);
```

```
      Serial.println("OFF");
    }
  } // Όταν έρθει μήνυμα που αφορά τον χρόνο (time).
  else if((String)topic == time_topic){      // Αν το μήνυμα είναι για το topic του time.
    if(isNumber(message) == 1){              // Ελέγχει αν είναι αριθμός.
      int timer = atoi(message.c_str());
      // Αν η τιμή που λάβαμε είναι μεγαλύτερη του 1 τότε προχωράμε σε αλλαγή του
//interval.
      if(timer >= 1){
        interval = 60*1000*long(timer);
        Serial.print("Changing data transmission time to ");
        Serial.print(interval);
        Serial.println(" millisecond.");
      }
    }
  }
}
//-------------------------------------------------------------------------------

//-----------------------------ntpCONNECT-------------------------------
  void ntpConnect(){
   // Αρχικοποίηση των στοιχείων για σύνδεση με τον NTP server και λήψη της χρονικής
//σήμανσης.
    configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
    struct tm timeInfo;
    if(!getLocalTime(&timeInfo)){        // Λαμβάνω την ημερομηνία από τον RTC.
     Serial.println("Failed to obtain time");
      return;
    }
    Serial.println(&timeInfo, "%A, %d %B %Y %H:%M:%S");
  }
//-----------------------------------------------------------------------
```

```
//------------------------------isNumber---------------------------------
int isNumber(String s){
  for(int i = 0;i<strlen(s.c_str());i++){  // Ελέγχει ένα ένα τα στοιχεία του string
    if(isdigit(s[i]) == 0){
      return 0;
    }
  }
  return 1;                          // Όλα τα ψηφία είναι αριθμητικοί χαρακτήρες από (0-9).
}
//-----------------------------------------------------------------------
```

Παράρτημα Β: ΕΓΚΑΤΑΣΤΑΣΗ ΛΟΓΙΣΜΙΚΩΝ ΔΙΑΚΟΜΙΣΤΗ RASPBERRY PI 4

Η διαδικασία ξεκινά με την εγκατάσταση ενός λειτουργικού συστήματος στην πλακέτα του raspberry pi 4. Θα εγκατασταθεί το λειτουργικό σύστημα, Raspberry Pi OS (παλαιότερα γνωστό ως "Raspbian"). Προκειμένου να προχωρήσει η εγκατάσταση θα χρειαστεί μια microSD Card, ένα Card Reader και το λογισμικό Raspberry Pi Imager, το οποίο θα εγκατασταθεί στον Η/Υ και θα το προμηθευτούμε από την επίσημη ιστοσελίδα του raspberry pi «https://www.raspberrypi.org/software/».

Τοποθετούμε το Card Reader με την microSD στον Η/Υ και ανοίγουμε την εφαρμογή Raspberry Pi Imager. Αφού διαλέξουμε το επιθυμητό λειτουργικό και την τοποθεσία που επιθυμούμε να το εγκαταστήσουμε, επιλέγουμε «write» και αρχίζει η εγκατάσταση στην microSD. Συνήθως αν χρεισιμοποιείται το raspberry pi ως διακομιστή επιλέγονται οι εκδόσεις Raspberry Pi OS ή Raspberry Pi OS Lite.

Εικόνα 52 Raspberry Pi Imager

Αφού τελειώσει η εγκατάσταση της επιθυμητής έκδοσης, ανοίγουμε τον κατάλογο αποθήκευσης της κάρτας microSD και δημιουργούμε ένα κενό αρχείο κειμένου (txt). Το αποθηκεύουμε με το όνομα «ssh», χωρίς να προσθέσουμε κάποια κατάληξη στο τέλος. Αυτή η διαδικασία είναι χρήσιμη αν θέλουμε χρησιμοποιήσουμε το πρωτόκολλο επικοινωνίας SSH.

Εικόνα 53 Αρχείο ssh

Στην πρώτη εκκίνηση όταν το Raspberry Pi OS εντοπίσει το «ssh», θα ενεργοποιήσει αυτόματα το πρωτόκολλο SSH (Secure Socket Shell), το οποίο θα επιτρέπει την απομακρυσμένη πρόσβαση στο παράθυρο εντολών τερματικού του Raspberry Pi.

Αφού τελειώσει η δημιουργία του αρχείου ssh καθορίζεται ο τρόπος διασύνδεσης του Η/Υ με την πλακέτα raspberry pi 4, ώστε να υπάρχει επικοινωνία με αυτήν. Μπορεί να επιτευχθεί σύνδεση διαμέσου τοπικού δικτύου wifi/Ethernet ή άμεση διασύνδεση με τον Η/Υ διαμέσου ενός Ethernet καλωδίου.

Στην περίπτωση που θα επιλεγεί η διασύνδεση διαμέσου wifi θα πρέπει να δημιουργηθεί μέσα στον ριζικό κατάλογο εγκατάστασης της κάρτας microSD ένα πρόσθετο αρχείο κειμένου με την ονομασία «wpa_supplicant.conf». Σε αυτό το αρχείο θα πρέπει να οριστούν οι απαραίτητοι παράμετροι για την σύνδεση της πλακέτας με το τοπικό δίκτυο wifi. Οι απαραίτητες εντολές είναι οι εξής:

```
country = GR
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config = 1
network = {
  scan_ssid = 1
  ssid ="wifi_ssid"
```

```
  psk ="wifi_password"
}
```

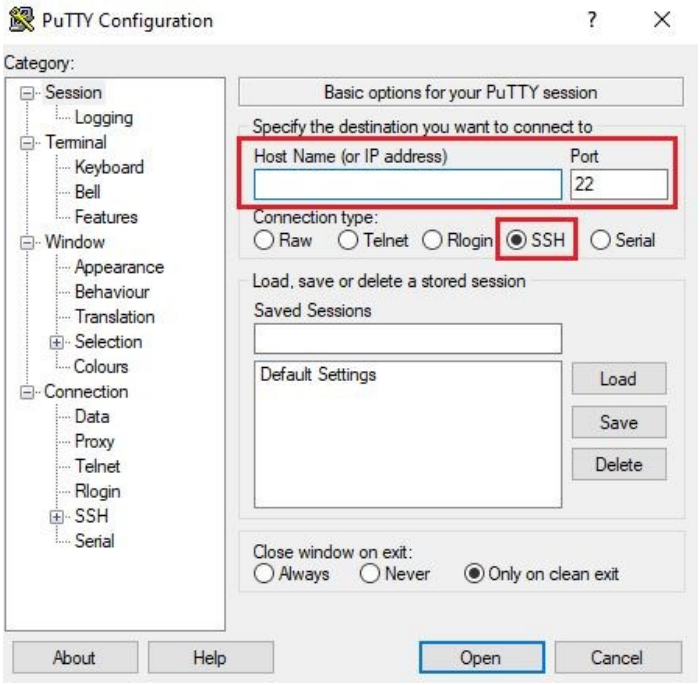
Για την συγκεκριμένη εργασία το raspberry pi 4 συνδέθηκε απευθείας στο δρομολογητή (router) διαμέσου ενός καλωδίου Ethernet. Μόλις εξασφαλιστεί ο τρόπος σύνδεσης του raspberry pi 4 με τον δρομολογητή, τοποθετείται η κάρτα microSD και ξεκινάμε την λειτουργία του.

Σύνδεση μέσω SSH

Προκειμένου να επιτευχθεί σύνδεση με το raspberry pi διαμέσου SSH, πρέπει να πραγματοποιηθεί λήψη και εγκατάσταση στον Η/Υ του λογισμικού Putty. Το οποίο μπορούμε να το προμηθευτούμε από το «https://www.putty.org/». Το Putty είναι ο κορυφαίος πελάτης SSH για Windows.

Μόλις τελειώσει η εγκατάσταση ανοίγουμε το λογισμικό putty ως administrator και συμπληρώνουμε τις απαιτούμενες ρυθμίσεις προκειμένου να προχωρήσει η διασύνδεση SSH με το raspberry pi. Στα αντίστοιχα πλαίσια επιλέγουμε SSH για τον πρωτόκολλο σύνδεσης, εισάγουμε την διεύθυνση ip του raspberry pi και την πόρτα 22. Αν δεν γνωρίζουμε την διεύθυνση ip που έχει δοθεί από το router στο raspberry pi, αντί γι αυτήν μπορούμε να συμπληρώσουμε το συγκεκριμένο πλαίσιο με τα ονόματα «raspberrypi» ή «raspberrypi.local». Στο router έχει καταχωρηθεί αυτόματα το όνομα της συσκευής και μας δρομολογεί στο raspberry pi 4, όταν γραφτεί ως διεύθυνση μία από τις προηγούμενες εντολές.

(Στην πρώτη προσπάθεια σύνδεσης διαμέσου του putty, θα εμφανιστεί μια προειδοποίηση ασφαλείας. Πατάμε "Yes" αφού συνδεόμαστε στο δικό μας Raspberry Pi)



Εικόνα 54 Putty

Αργότερα εμφανίζει το παράθυρο εντολών του raspberry pi όπου γράφουμε το username και password. Την πρώτη φορά που θα συνδεθούμε στο raspberry pi θα χρησιμοποιήσουμε ως username το «pi» και ως κωδικό πρόσβασης το «raspberry».

Σε περίπτωση που επιθυμούμε να αλλάξουμε το κωδικό πρόσβασης για να δώσουμε παραπάνω ασφάλεια στην συσκευή μας πληκτρολογούμε την εντολή **passwd** και συμπληρώνουμε τα στοιχεία που μας ζητούνται.

Αφού συνδεθούμε στο παράθυρο εντολών του raspberry pi η πρώτη διαδικασία που θα πραγαμτοποιηθεί είναι να αναβαθμίσουμε το λειτουργικό της πλακέτας στην πιο πρόσφατη έκδοση, προκειμένου να περιέχει τα πιο πρόσφατα πακέτα. Αυτή η διαδικασία πραγματοποιείται γράφοντας στο παράθυρο εντολών, τις εντολές:

sudo apt update
sudo apt upgrade

(Το apt update ενημερώνει τη λίστα των διαθέσιμων πακέτων και των εκδόσεών τους, αλλά δεν εγκαθιστά ή αναβαθμίζει κανένα πακέτο. Το apt upgrade εγκαθιστά στην πραγματικότητα νεότερες εκδόσεις των πακέτων. Μετά την ενημέρωση των λιστών, ο διαχειριστής πακέτων γνωρίζει για τις διαθέσιμες ενημερώσεις του λογισμικού που έχει εγκατασταθεί. Αυτός είναι και ο λόγος που πρώτα πρέπει να γίνει η εντολή update)

Κατά την διαδικασία της ενημέρωσης θα εμφανιστεί μια ερώτηση στην οποία γράφουμε «Y». Μόλις τελειώσει η διαδικασία καλό είναι να γίνει επανακίνηση στην συσκευή. Αυτό θα το πετύχουμε γράφοντας στο παράθυρο εντολών την εντολή:

sudo reboot

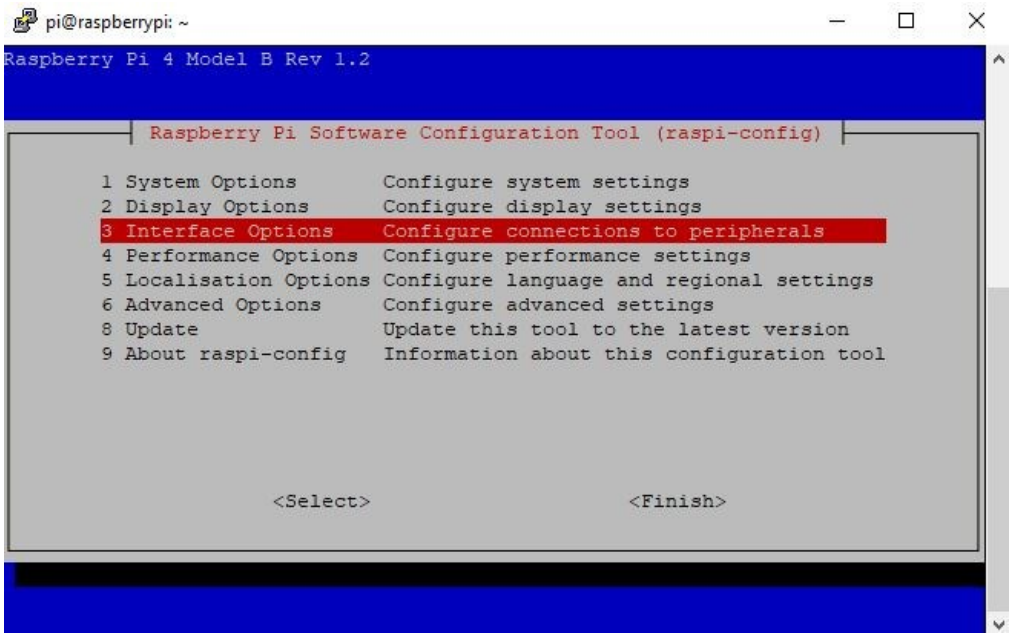
Πλέον το Raspberry Pi διαθέτει μια βασική εγκατάσταση του Raspberry Pi OS, έτοιμο για προσαρμογή και εγκατάσταση των επιθυμητών λογισμικών.

Ενεργοποίηση και σύνδεση μέσω VNC:

Αν στο raspberry pi έχει εγκατασταθεί κάποια από τις desktop εκδόσεις και όχι η lite τότε μπορούμε να συνδεθούμε στην πλακέτα διαμέσου Virtual Network Computing (VNC), προκειμένου να έχουμε πρόσβαση στο γραφικό περιβάλλον του raspberry pi χρησιμοποιόντας έναν απομακρυσμένο Η/Υ ή smartphone. Για να ενεργοποιήσουμε την συγκεκριμένη ρύθμιση πρέπει να μεταβούμε στο παράθυρο εντολών του raspberry pi διαμέσου της εφαρμογής putty και να γράψουμε την εντολή:

sudo raspi-config

Στο παράθυρο που θα εμφανιστεί επιλέγουμε «Interfacing Options» και αργότερα διαλέγουμε το «VNC» από την εμφανιζόμενη λίστα. Μόλις ολοκληρώσουμε την διαδικασία επιλέγουμε «Yes» και πλέον έχουμε ενεργοποιήσει την ρύθμιση VNC και όλες τις δυνατότητες που αυτή μας προσφέρει.



Εικόνα 55 Ενεργοποίηση VNC στο Raspberry Pi

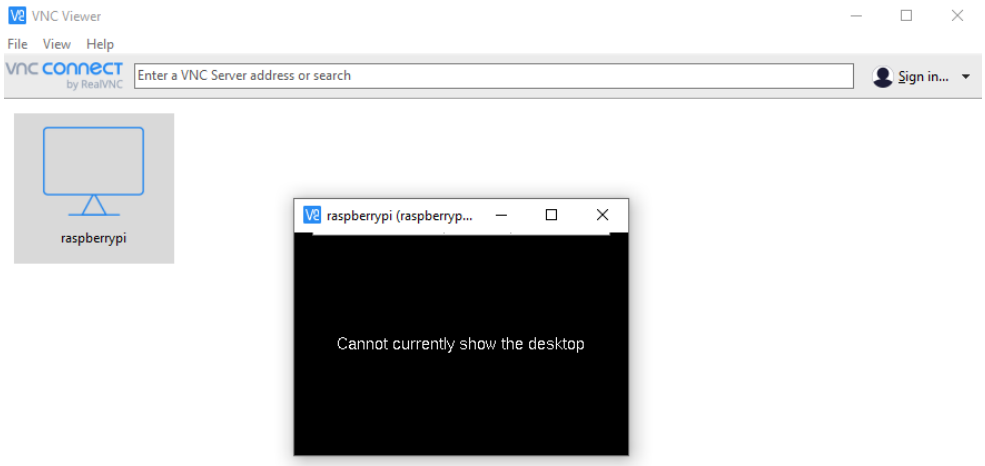
Διαδικασία στον υπολογιστή:

Προτού γίνει δυνατή η σύνδεσή με την πλακέτα raspberry pi θα πρέπει να εγκαταστήσουμε το λογισμικό VNC Viewer στον Η/Υ. Το VNC Viewer μπορούμε να το προμηθευτούμε δωρεάν από την ιστιοσελίδα «https://www.realvnc.com/en/connect/download/viewer/».

Μόλις τελειώσει η εγκατάσταση του VNC Viewer και εκτελέσουμε το λογισμικό διαλέγουμε από το «File menu» την επιλογή «New connection». Στο παράθυρο που θα εμφανιστεί πληκτρολογούμε στο πεδίο «VNC Server» την διεύθυνση ip του raspberry pi ή γράφουμε τα ονόματα «raspberrypi» ή «raspberry.local». Χωρίς να αλλάξουμε τίποτα άλλο επιλέγουμε «OK» και πλέον θα έχει εμφανιστεί το εικόνίδιο σύνδεσης με την πλακέτα.

Πατώντας διπλό κλικ στο εικονίδιο σύνδεσης θα μας ζητηθεί να εισάγουμε το όνομα χρήστη και τον κωδικό πρόσβασης που δώσαμε στο raspberry pi. Την πρώτη φορά που θα συνδεθούμε θα εμφανιστεί μια προειδοποίηση ασφαλείας, επιλέγουμε το OK και θα προχωρήσει η σύνδεση κανονικά. Πλέον έχουμε πρόσβαση στο γραφικό περιβάλλον του λειτουργικού συστήματος της πλακέτας και μπορούμε να το χειριστούμε από απόσταση.

Σε πολλές περιπτώσεις μπορεί να μην είναι δυνατή η σύνδεση με το γραφικό περιβάλλον της πλακέτα και να εμφανίζεται το μήνυμα «Cannot currently show the desktop».

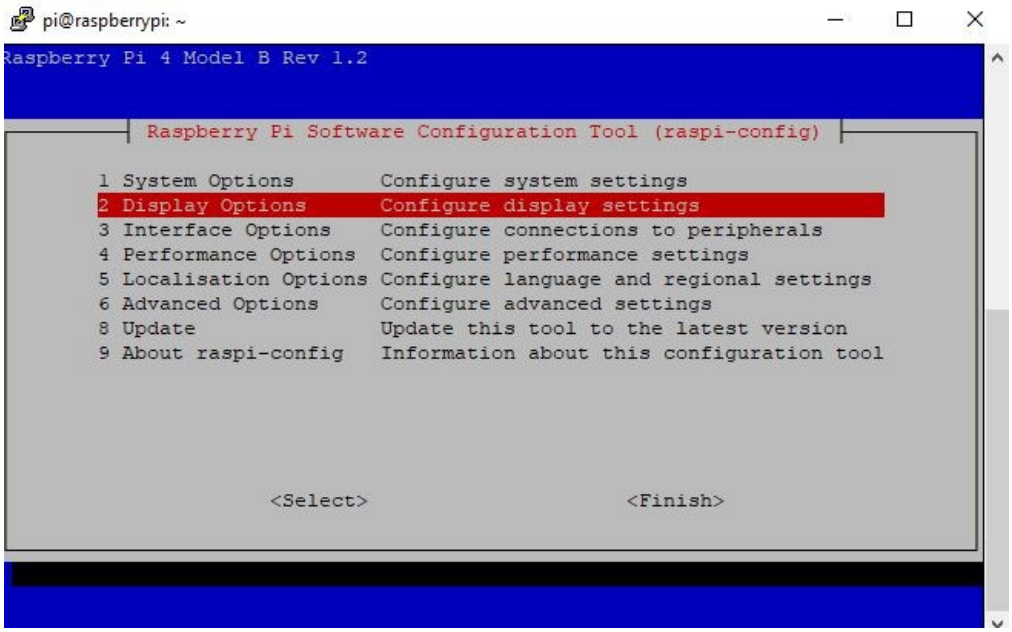


Εικόνα 56 Αδυναμία σύνδεσης VNC με το Raspberry Pi

Σε αυτή την περίπτωση θα πρέπει να αλλάξουμε το Screen Resolution στο Raspberry Pi μπαίνοντας στο μενού των ρυθμίσεων με την εντολή:

sudo raspi-config

Από το παράθυρο που θα εμφανίστεί επιλέγουμε «Display Options» και αργότερα «Resolution». Από την εμφανιζόμενη λίστα επιλέγουμε την ρύμθιση που αντιστοιχεί στην οθόνη υπολογιστή που χρησιμοποιούμε και βγαίνουμε από τις ρυθμίσεις. Για να εφαρμοστεί η επιλογή θα πρέπει να γίνει επανακίνηση στο raspberry pi, οπότε γράφουμε την εντολή «sudo reboot». Ετσι την επόμενη φορά που θα δοκιμάσουμε να συνδεθούμε στην πλακέτα διαμέσου VNC, θα εμφανιστεί κανονικά το γραφικό περιβάλλον.



Εικόνα 57 Αλλαγή Ρυθμίσεων Ανάλυσης Οθόνης

Εγκατάσταση Mosquitto MQTT broker

Έχοντας τελειώσει με όλες τις απαραίτητες ρυθμίσεις για απομακρυσμένο έλεγχο της πλακέτας και αφού έχουμε εγκαταστήσει το επιθυμητό λειτουργικό σύστημα, είμαστε έτοιμοι να προχωρήσουμε στην εγκατάσταση του απαραίτητου λογισμικού για την αποθήκευση και διαχείρηση των δεδομένων του συστήματος που θα δημιουργήθει.

Αρχικά θα εγκατασταθεί το Mosquitto MQTT broker προκειμένου να είναι εφικτή η διαχείρηση των δεδομένων διαμέσου του πρωτοκόλλου MQTT από την πλακέτα του raspberry pi.

Για να εγκαταστήσουμε το λογισμικό του Mosquitto MQTT πρέπει να εισαχθούμε διαμέσου του Putty, στο παράθυρο εντολών του Raspberry Pi και να γράψουμε τις εντολές:

sudo apt update

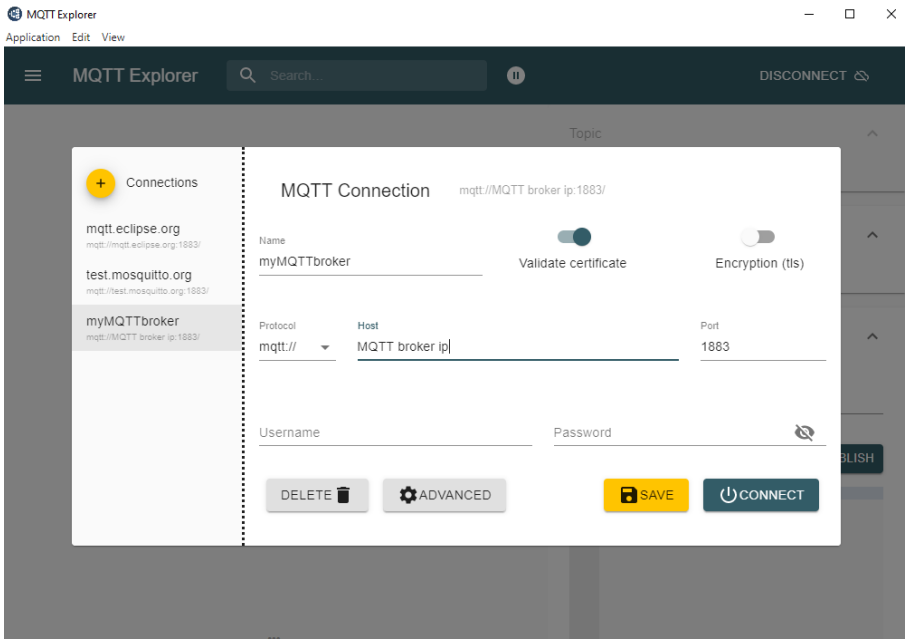
sudo apt install -y mosquitto mosquitto-clients

Προκειμένου να ενεργοποιείται το Mosquitto MQTT broker αυτόματα με την ενεργοποίηση ή επανεκκίνηση του raspberry pi γράφουμε την εντολή:

sudo systemctl enable mosquitto.service

Έλεγχος λειτουργίας MQTT broker:

Αφού ολοκληρωθεί η εγκατάσταση του borker θα εγκαταστήσουμε στον προσωπικό υπολογιστή ένα λογισμικό παρατήρησης και ελέγχου της σωστής λειτουργίας του Server. Πρόκειται για το λογισμικό MQTT Explorer που είναι ένας ολοκληρωμένος πελάτης MQTT και βοηθάει στο να έχουμε μια εύκολη παρατήρηση της λειτουργίας του MQTT server. Το λογισμικό μπορούμε να το προμηθευτούμε από το «http://mqtt-explorer.com/».



Εικόνα 58 MQTT Explorer

Η λειτουργία του MQTT Explorer είναι πολύ απλή, δημιουργούμε μια νέα σύνδεση δίνοντάς της το όνομα που επιθυμούμε. Σε αυτήν ορίζουμε την διεύθυνση ip του raspberry pi και την πόρτα 1883 η οποία αναφέρεται στον mosquitto MQTT broker. Αργότερα επιλέγουμε connection και αν όλα λειτουργούν σωστά έχουμε συνδεθεί στον broker και έχουμε γραφτεί σε όλα τα topic που διαθέτει.

Αν δεν γνωρίζουμε την διεύθυνση IP γράφουμε **hostname -I** στην γραμμή εντολών του raspberry pi.

Εγκατάσταση Node-RED

Προτού εγκαταστήσουμε το Node-RED χρήσιμο είναι να κάνουμε ενημέρωση στο raspberry pi έτσι ώστε να ενημερωθούν τα υπάρχοντα πακέτα. Την συγκεκριμένη

διαδικασία μπορούμε να την παραλείψουμε αν η εγκατάσταση του Node-Red γίνει την ίδια μέρα που κάναμε εγκατάσταση στο λειτουργικό σύστημα της πλακέτας, μιας και έχουμε προβεί ήδη στην συγκεκριμένη διαδικασία. Οπότε γράφουμε τις εντολές:

sudo apt update

sudo apt upgrade

Αρχικά πριν προχωρήσουμε στην εγκατάσταση του Node-Red θα εγκαταστήσουμε το NPM. Το NPM είναι ένας διαχειριστής πακέτων για την πλατφόρμα JavaScript Node.js. Τοποθετεί και διαχειρίζεται έξυπνα τις διάφορες λειτουργικές μονάδες Node.js έτσι ώστε να μπορούν εύκολα να βρεθούν και να χρησιμοποιηθούν. Είναι χρήσιμο και απαραίτητο εργαλείο κυρίως αν σκοπεύουμε να εγκαταστήσουμε πρόσθετους κόμβους (nodes) στο Node-Red, προκειμένου να αυξήσουμε την παλέτα λειτουργιών που διαθέτει. Για να εγκαταστήσουμε το "build-essential" εκτελούμε την εντολή:

sudo apt install build-essential git

Τώρα μπορούμε να εκτελέσουμε το επίσημο script εγκατάστασης του Node-RED. Αυτή η εντολή θα εγκαταστήσει μαζί με το Node-Red και το πακέτο Node.js αν δεν το έχουμε ήδη εγκατεστημένο στην πλακέτα Raspberry Pi. Για να ξεκινήσει η εγκατάσταση γράφουμε την εντολή:

bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)

Κατά την διάρκεια της εγκατάστασης θα εμφανιστούν δύο ερωτήσεις, γράφουμε και στις δύο «y» και αφήνουμε να συνεχιστεί η διαδικασία της εγκατάστασης.

Όπως ορίσαμε στο Mosquitto MQTT, θέλουμε το Node-RED να ενεργοποιείται αυτόματα με την ενεργοποίηση ή επανεκκίνηση του raspberry pi. Για να ρυθμίσουμε την συγκεκριμένη λειτουργία γράφουμε την εντολή:

sudo systemctl enable nodered.service

Αντίθετα όμως με το Mosquitto MQTT που μετά την εγκατάσταση αρχίζει αυτόματα η εκτέλεση του λογισμικού, με το Node-Red θα πρέπει να κάνουμε επανακίνηση την πλακέτα ή να εκτελέσουμε εντολή ενεργοποίησης του λογισμικού. Την συγκεκριμένη διαδικασία δεν χρειάζεται να την επαναλάβουμε στο μέλλον μιας και έχουμε καθορίσει την αυτόματη

ενεργοποίηση του Node-Red με την ενεργοποίηση της πλακέτας. Για να ξεκινήσουμε το Node-RED ως υπηρεσία χειροκίνητα αυτή τη φορά, θα γράψουμε την εντολή:

sudo systemctl start nodered.service

Άνοιγμα του Node-RED Editor:

Έχοντας τελειώσει την εγκατάσταση του Node-Red, Θα πρέπει να ελέξουμε την σωστή λειτουργία του, μπαίνοντας στο περιβάλλον προγραμματισμού που διαθέτει. Σε ένα πρόγραμμα περιήγησης μεταβαίνουμε στη διεύθυνση «http: // <ip address>: 1880», όπου η διεύθυνση IP είναι η διεύθυνση του Raspberry Pi και το 1880 είναι η πόρτα στην οποία ακούει το Node-Red. Οπότε στο πρόγραμμα περιήγησης που χρησιμοποιούμε γράφουμε την εντολή:

http: // <ip address>: 1880

Στην περίπτωση που θέλουμε να μεταβούμε στο γραφικό περιβάλλον του Node-Red από την ίδια πλατφόρμα στην οποία το έχουμε εγκαταστήσει, μπορούμε να το ανοίξουμε από τη διεύθυνση: **http: // localhost: 1880**.

Εγκατάσταση πρόσθετων στο Node-Red:

Το Node-RED μπορεί να επεκταθεί εγκαθιστώντας πρόσθετες μονάδες που του προσφέρουν επιπλέον δυνατότητες. Αυτή η λειτουργία είναι απαραίτητη προκειμένου να επιτευχθεί σύνδεση με την βάση δεδομένων InfluxDB και η δυνατότητα δημιουργίας Dashboard.

Υπάρχουν δύο τρόποι για να εγκατασταθούν πρόσθετα script στο Node-Red.

Ο 1ος τρόπος και αυτόν που θα χρησιμοποιήσουμε είναι μέσα από το περιβάλλον χρήσης «http: // <ip address>: 1880». Από εκεί μεταβηβαζόμαστε στην πάνω δεξιά γωνία και επιλέγουμε το «**εικονίδιο με τις τρεις γραμμές»,** αργότερα επιλέγουμε το **«Manage Palette»** και τέλος διαλέγουμε το tab με το όνομα **«Install»**

Στο παράθυρο που θα εμφανιστεί πληκτρολογούμε στην αναζήτηση τα επιθυμητά πακέτα. Στην συγκεκριμένη περίπτωση θα εγκαταστήσουμε το «node-red-contrib-influxdb» και το «node-red-dashboard».

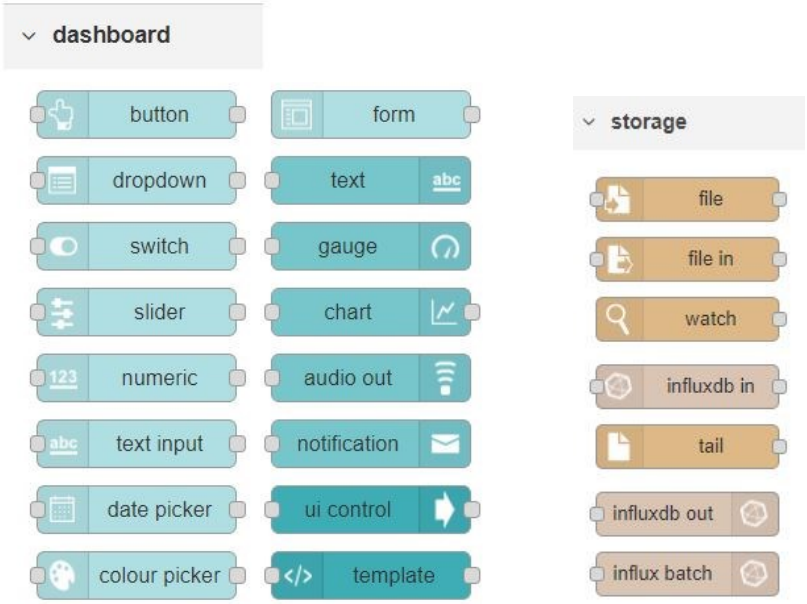
Εναλλακτικός 2ος τρόπος είναι μέσα από το παράθυρο εντολών του raspberry pi. Εκεί γράφουμε τις εντολές:

cd ~/.node-red

npm install node-red-dashboard

npm install node-red-contrib-influxdb

Αν οι συγκεκριμένες εντολές δεν γραφτούν όταν βρισκόμαστε στο cd ~/.node-red δεν θα μπορέσουμε να εγκαταστήσουμε τα πακέτα στο σωστό μέρος και δεν θα είναι δυνατή η λειτουργικότητά τους.



Εικόνα 59 Dashboard-Storage Palette

Εγκατάσταση InfluxDB:

Για την εγκατάσταση του InfluxDB θα χρησιμοποιηθεί το επίσημο αποθετήριο της εταιρίας. Θα προσθέσουμε το κλειδί αποθετηρίου InfluxDB στο Raspberry Pi, με το οποίο θα επιτρέψουμε στον διαχειριστή πακέτων του λειτουργικού συστήματος να ψάξει στο αποθετήριο και να επαληθεύσει τα πακέτα που εγκαθιστά.

Μπορούμε να προσθέσουμε το κλειδί InfluxDB εκτελώντας την ακόλουθη εντολή:

wget -qO- https://repos.influxdata.com/influxdb.key | sudo apt-key add -

Αφού εγκατασταθεί το κλειδί αποθετηρίου InfluxDB στο Raspberry Pi, θα προχωρήσουμε και θα προσθέσουμε το αποθετήριο στη λίστα πηγών. Υπάρχουν διάφορες διαθέσιμες εκδόσεις, για να μάθουμε ποια έκδοση χρησιμοποιούμε θα πληκτρολογήσουμε την εντολή:

lsb_release -a

Αυτή η διαδικασία είναι απαραίτητη επειδή ανάλογα με το ποια έκδοση Raspbian διαθέτουμε θα χρησιμοποιηθεί διαφορετική εντολή εγκατάστασης. Στην συγκεκριμένη περίπτωση θα εγκαταστήσουμε την έκδοση που απευθύνεται στο Raspbian Buster.

Για το **Raspbian Buster** γράφουμε την εντολή:

echo "deb https://repos.influxdata.com/debian buster stable" | sudo tee /etc/apt/sources.list.d/influxdb.list

Εναλλακτικά για το **Raspbian Stretch** γράφουμε την εντολή:

echo "deb https://repos.influxdata.com/debian stretch stable" | sudo tee /etc/apt/sources.list.d/influxdb.list

Με την προσθήκη του νέου αποθετηρίου χρειάζεται να ενημερωθεί ξανά η λίστα πακέτων, έτσι ώστε ο διαχειριστής πακέτων να κάνει αναζήτηση στο αποθετήριο που μόλις προσθέθηκε. Το λειτουργικό σύστημα δεν το κάνει αυτό αυτόματα με την προσθήκη της νέας λίστας, οπότε θα πρέπει να γράψουμε στο Raspberry Pi την εντολή:

sudo apt update

Έχοντας ενημερώσει το Raspberry Pi για τα νέα πακέτα, μπορούμε να προχωρήσουμε στην εγκατάσταση του InfluxDB. Γράφουμε την εντολή εκτελούμε την εντολή:

sudo apt install -y influxdb

Προκειμένου να ενεργοποιείται το InfluxDB αυτόματα με την ενεργοποίηση ή επανεκκίνηση του raspberry pi πρέπει να ενημερωθεί ο διαχειριστής υπηρεσιών systemctl. Αυτό επιτυγχάνεται με τις εντολές:

sudo systemctl unmask influxdb.service

sudo systemctl enable influxdb.service

Όπως με το Node-Red έτσι και εδώ για να ξεκινήσουμε τον διακομιστή InfluxDB την πρώτη φορά μετά την εγκατάσταση θα πρέπει να εκτελέσουμε την εντολή:

sudo systemctl start influxdb

Πλέον όποτε ενεργοποιείται το raspberry pi 4 θα εκτελείται και ο διακομιστής influxDB, αυτόματα χωρίς να χρειάζεται να κάνουμε κάποια διαδικασία.

Ορισμός ρυθμίσεων και δημιουργία της βάσης δεδομένων στο influxDB:

Για να μεταφερθούμε στο περιβάλλον διαχείρισης της βάσης δεδομένων InfluxDB γράφουμε την εντολή:

influx

Στη συνέχεια θα δημιουργήσουμε μια βάση δεδομένων όπου θα αποθηκεύσουμε τα δεδομένα των αισθητήρων. Για την συγκεκριμένη εργασία θα δημιουργήσουμε μια βάση δεδομένων με το όνομα «thermostat». Γράφουμε την εντολή:

CREATE DATABASE thermostat

Λόγω του τρόπου με τον οποίο λειτουργεί το InfluxDB, δεν χρειάζεται να δημιουργήσουμε ένα σχήμα με πίνακες και στήλες, όπως θα γινόταν σε κάποια άλλη σχεσιακή βάση δεδομένων, π.χ. MariaDB, MySQL, Postgres ή SQL Server. Το μόνο που χρειάζεται να κάνουμε είναι να δημιουργήσουμε μια κενή βάση δεδομένων και θα συμπληρωθεί αυτόματα όταν αρχίσουμε να στέλνουμε δεδομένα σε αυτήν.

Αυτή η ιδιότητα δεν αναιρεί την δυνατότητα τροποποιήσεων στην βάση δεδομένων InfluxDB. Προκειμένου να προβούμε σε τροποποιήσεις πρέπει να εισέλθουμε μέσα στην συγκεκριμένη βάση χρησιμοποιώντας την εντολή:

USE thermostat

Στην συνέχεια θα δημιουργήσουμε ένα χρήστη που θα του δοθεί πλήρης πρόσβαση στην βάση δεδομένων. Στον συγκεκριμένο χρήστη θα δώσουμε δυνατότητες διαχειριστή με τις εντολές:

CREATE USER <username> WITH PASSWORD '<password>' WITH ALL PRIVILEGES

GRANT ALL PRIVILEGES ON thermostat TO <username>

Προσοχή ο κωδικός πρέπει να γραφτεί μέσα σε ‘’ αλλίως θα βγεί μήνυμα σφάλματος. Δεν πρέπει να κάνουμε την συγκεκριμένη διαδικασία για το username.

Για να ελένξουμε και να δούμε την λίστα όλων των υπαρχόντων χρηστών χρησιμοποιείται η εντολή:

SHOW USERS

Για έξοδο από το περιβάλλον διαχείρισης client Influx πληκτρολογούμε την εντολή: **exit**

Εγκατάσταση Grafana:

Όπως και με το InfluxDB, θα εγκαταστήσουμε το Grafana προσθέτοντας το επίσημο αποθετήριο. Θα προσθέσουμε το κλειδί αποθετηρίου Grafana στο Raspberry Pi, με το οποίο θα επιτρέψουμε στον διαχειριστή πακέτων του λειτουργικού συστήματος να ψάξει στο αποθετήριο και να επαληθεύσει τα πακέτα που εγκαθιστά. Το κλειδί APT θα προστεθεί με την εντολή:

wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -

Με το κλειδί μπορεί να προστεθεί το αποθετήριο Grafana στη λίστα των πηγών στα πακέτα που περιέχει το Raspberry Pi. Η προσθήκη του αποθετηρίου στην λίστα γίνεται με την εντολή:

echo "deb https://packages.grafana.com/oss/deb stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list

Με την προσθήκη του αποθετηρίου χρειάζεται να ενημερωθεί ξανά η λίστα πακέτων. Έτσι ώστε ο διαχειριστής να κάνει αναζήτηση στο αποθετήριο που μόλις προσθέσαμε για νέα πακέτα. Οπότε εκτελούμε την εντολή ενημέρωσης:

sudo apt update

Αφού πλέον το Raspberry Pi αναγνωρίζει το Grafana, προχωράμε στην εγκατάσταση της τελευταίας έκδοσης με την εντολή:

sudo apt install -y grafana

Προκειμένου να ενεργοποιείται το Grafana αυτόματα με την ενεργοποίηση ή επανεκκίνηση του raspberry pi πρέπει να ενημερωθεί ο διαχειριστής υπηρεσιών systemctl. Αυτό επιτυγχάνεται με τις εντολές:

sudo systemctl unmask grafana-server.service

sudo systemctl enable grafana-server.service

Για να ξεκινήσουμε το Grafana την πρώτη φορά μετά την εγκατάσταση θα πρέπει να εκτελέσουμε την εντολή:

sudo systemctl start grafana-server

Έλεγχος λειτουργίας του Grafana:

Για να επιτευχθεί σύνδεση με το γραφικό περιβάλλον του Grafana, καθώς και ο έλεγχος της λειτουργίας του ανοίγουμε ένα πρόγραμμα περιήγησης και πληκτρολογούμε την διεύθυνση IP του raspberry pi καθώς και την πόρτα 3000 στην οποία ορίζεται το Grafana. Οπότε στο πρόγραμμα περιήγησης που χρησιμοποιούμε γράφουμε την εντολή:

http: // <ip address>: 3000

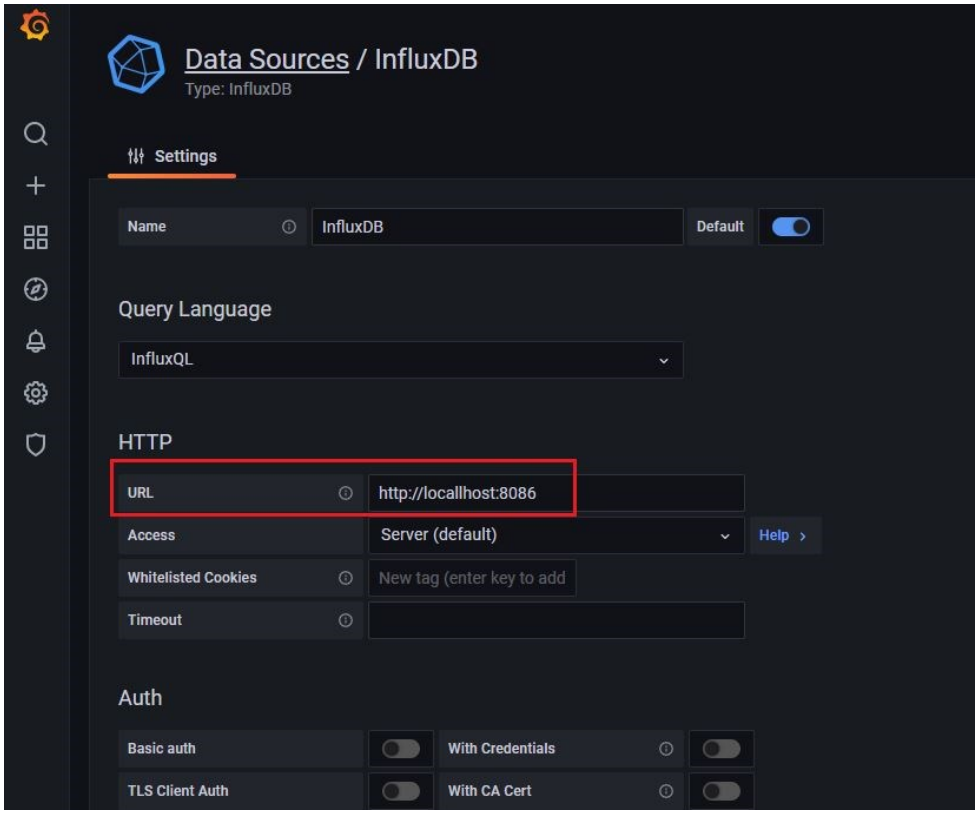
Στην περίπτωση που θέλουμε να μεταβούμε στο γραφικό περιβάλλον του Grafana από την ίδια πλατφόρμα στην οποία το έχουμε εγκαταστήσει, μπορούμε να το ανοίξουμε από τη διεύθυνση: **http: // localhost: 3000**.

Την πρώτη φορά που θα συνδεθούμε στη Grafana θα χρησιμοποιήσουμε ως username το «**admin**» και ως κωδικό πρόσβασης το «**admin**». Αργότερα θα ζητηθεί να αλλάξουμε τον κωδικό πρόσβασης του διαχειριστή. Αφού βάλουμε τον κωδικό που προτειμάμε επιλέγουμε «save» και εισερχόμαστε στο περιβάλλον χρήστη.

Το πρώτο πράγμα που πρέπει να κάνουμε στο γραφικό περιβάλλον του Grafana είναι να δημιοργήσουμε μια σύνδεση μεταξύ της βάσης δεδομένων «thermostat» που δημιουργήσαμε στο InfluxDB και του Grafana. Αυτή η διαδικασία είναι απαραίτητη προκειμένου να μπορούμε να δεχόμαστε δεδομένα προς οπτικοποίηση με το Grafana.

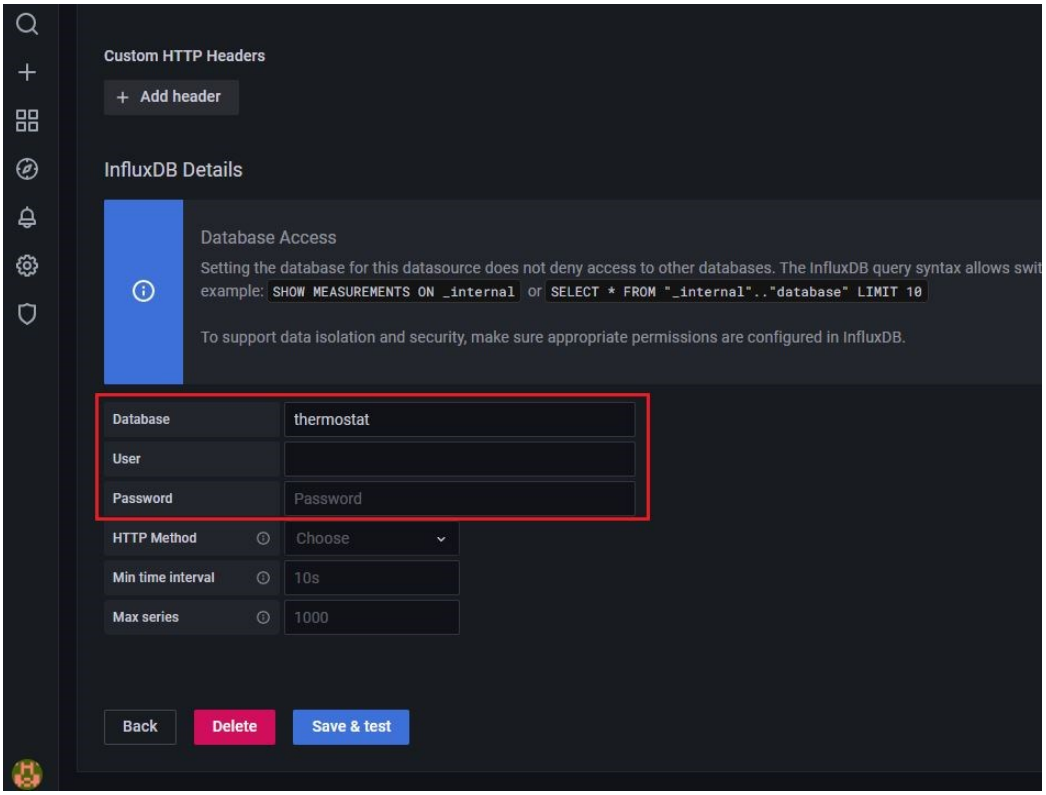
Από την αρχική σελίδα επιλέγουμε το «Data Sources» το οποίο θα μας εμφανίσει μια λίστα με όλες τις εταιρίες διακομιστών που υποστηρίζονται από το Grafana. Βρίσκουμε και επιλέγουμε την «InfluxDB» που βρήσκεται στην κατηγορία «Timeseries Databases».

Προκειμένου να κάνουμε εφικτή την διασύνδεση θα χρειαστεί να συμπληρώσουμε μερικές ρυθμίσεις. Αρχικά θα ορίσουμε την διεύθυνση της πλακέτας που είναι εγκατεστημένος ο InfluxDB διακομιστής στο πεδίο «URL». Καθώς εκτελούμε και τις δύο υπηρεσίες στο ίδιο Raspberry Pi, θα γράψουμε ως διεύθυβση το «http://localhost:8086». Το 8086 είναι η πόρτα που ακούει το InfluxDB.



Εικόνα 60 Ορισμός διεύθυνσης βάσης InfluxDB

Στη συνέχεια πρέπει να ορίσουμε το όνομα της βάση δεδομένων, τον χρήστη και τον κωδικό πρόσβασης που ορίσαμε νωρίτερα, προκειμένου να γίνει δυνατή η διασύνδεση με το InfluxDB.



Εικόνα 61 Ρυθμίσεις Συνδεσιμολογίας Με InfluxDB

Τέλος επιλέγουμε « Save & Test» και αποθηκεύουμε τις ρυθμίσεις. Πλέον όποτε επιλέγουμε την βάση InfluxDB θα συνδεόμαστε αυτόματα στον «thermostat» και θα οπτικοποιούμε τις μετρήσεις των αισθητήρων που έχουν καταχωρηθεί.

Παράρτημα Γ: ΚΩΔΙΚΑΣ ΣΤΟ NODE-RED ΣΕ ΜΟΡΦΗ COMPACT JSON

[{"id":"a2a9f2ae6b048ed5","type":"tab","label":"Inside Sensors","disabled":false,"info":""},{"id":"b1ab34a6f7efaa74","type":"tab","label":"Outside Sensors","disabled":false,"info":""},{"id":"1e1d5d4fe0c4f0d6","type":"tab","label":"Boiler Room","disabled":false,"info":""},{"id":"a5b7d3d6cf06605c","type":"tab","label":"Additional Functions","disabled":false,"info":""},{"id":"215adc636c610a74","type":"mqtt-broker","name":"","broker":"localhost","port":"1883","clientid":"","usetls":false,"protocolVersion":"4","keepalive":"60","cleansession":true,"birthTopic":"","birthQos":"0","birthPayload":"","birthMsg":{},"closeTopic":"","closeQos":"0","closePayload":"","closeMsg":{},"willTopic":"","willQos":"0","willPayload":"","willMsg":{},"sessionExpiry":""},{"id":"efcb09695f40f060","type":"influxdb","hostname":"127.0.0.1","port":"8086","protocol":"http","database":"thermostat","name":"","usetls":false,"tls":"","influxdbVersion":"1.x","url":"http://localhost:8086","rejectUnauthorized":true},{"id":"8fd6091f8efdbff5","type":"ui_base","theme":{"name":"theme-dark","lightTheme":{"default":"#0094CE","baseColor":"#0094CE","baseFont":"-apple-system,BlinkMacSystemFont,Segoe UI,Roboto,Oxygen-Sans,Ubuntu,Cantarell,Helvetica Neue,sans-serif","edited":false},"darkTheme":{"default":"#097479","baseColor":"#097479","baseFont":"-apple-system,BlinkMacSystemFont,Segoe UI,Roboto,Oxygen-Sans,Ubuntu,Cantarell,Helvetica Neue,sans-serif","edited":true,"reset":false},"customTheme":{"name":"Untitled Theme 1","default":"#4B7930","baseColor":"#4B7930","baseFont":"-apple-system,BlinkMacSystemFont,Segoe UI,Roboto,Oxygen-Sans,Ubuntu,Cantarell,Helvetica Neue,sans-serif"},"themeState":{"base-color":{"default":"#097479","value":"#097479","edited":false},"page-titlebar-backgroundColor":{"value":"#097479","edited":false},"page-backgroundColor":{"value":"#111111","edited":false},"page-sidebar-backgroundColor":{"value":"#333333","edited":false},"group-textColor":{"value":"#0eb8c0","edited":false},"group-borderColor":{"value":"#555555","edited":false},"group-backgroundColor":{"value":"#333333","edited":false},"widget-textColor":{"value":"#eeeeee","edited":false},"widget-backgroundColor":{"value":"#097479","edited":false},"widget-borderColor":{"value":"#333333","edited":false},"base-font":{"value":"-apple-system,BlinkMacSystemFont,Segoe UI,Roboto,Oxygen-Sans,Ubuntu,Cantarell,Helvetica Neue,sans-serif"}},"angularTheme":{"primary":"indigo","accents":"blue","warn":"red","background":"grey","palette":"light"}},"site":{"name":"THERMOSTAT","hideToolbar":"false","allowSwipe":"false","lockMenu":"false","allowTempTheme":"true","dateFormat":"DD/MM/YYYY","sizes":{"sx":48,"sy":48,"gx":6,"gy":6,"cx":6,"cy":6,"px":0,"py":0}}},{"id":"514435e2b139ad59","type":"ui_tab","name":"ΚΕΝΤΡΙΚΗ ΣΕΛΙΔΑ","icon":"dashboard","order":1,"disabled":false,"hidden":false},{"id":"6eff3ecd4e79cb10","type":"ui_tab","name":"ΕΣΩΤΕΡΙΚΟΣ ΧΩΡΟΣ","icon":"dashboard","order":2,"disabled":false,"hidden":false},{"id":"905ff487abaf2841","type":"ui_tab","name":"ΕΞΩΤΕΡΙΚΟΣ

ΧΩΡΟΣ","icon":"dashboard","order":3,"disabled":false,"hidden":false},{"id":"82b494d49f26b61c","type":"ui_tab","name":"ΛΕΒΗΤΟΣΤΑΣΙΟ","icon":"dashboard","order":4,"disabled":false,"hidden":false},{"id":"42eff3bb8dea9da5","type":"ui_group","name":"ΕΣΩΤΕΡΙΚΗ
ΥΓΡΑΣΙΑ","tab":"514435e2b139ad59","order":2,"disp":true,"width":6,"collapse":false},{"id":"4724563832291ce5","type":"ui_group","name":"ΕΞΩΤΕΡΙΚΗ
ΘΕΡΜΟΚΡΑΣΙΑ","tab":"514435e2b139ad59","order":3,"disp":true,"width":6,"collapse":false},{"id":"969f4575dbd73cb8","type":"ui_group","name":"ΕΞΩΤΕΡΙΚΗ
ΥΓΡΑΣΙΑ","tab":"514435e2b139ad59","order":4,"disp":true,"width":6,"collapse":false},{"id":"a8b2374324598934","type":"ui_group","name":"ΕΣΩΤΕΡΙΚΗ
ΘΕΡΜΟΚΡΑΣΙΑ","tab":"514435e2b139ad59","order":1,"disp":true,"width":6,"collapse":false},{"id":"27ceab0bc7228cc3","type":"ui_group","name":"ΘΕΡΜΟΚΡΑΣΙΑ
ΣΩΜΑΤΟΣ","tab":"514435e2b139ad59","order":5,"disp":true,"width":6,"collapse":false},{"id":"e791fbc1d3197770","type":"ui_spacer","z":"a2a9f2ae6b048ed5","name":"spacer","group":"5652fd7ca1ffc7cb","order":5,"width":6,"height":1},{"id":"54602633d858ed4f","type":"ui_spacer","z":"a2a9f2ae6b048ed5","name":"spacer","group":"5652fd7ca1ffc7cb","order":6,"width":6,"height":1},{"id":"220d73d96f0167fd","type":"ui_spacer","z":"a2a9f2ae6b048ed5","name":"spacer","group":"5652fd7ca1ffc7cb","order":7,"width":6,"height":1},{"id":"aff7a381bca95830","type":"ui_spacer","z":"a2a9f2ae6b048ed5","name":"spacer","group":"5652fd7ca1ffc7cb","order":10,"width":6,"height":1},{"id":"4949e2e1caaa1988","type":"ui_spacer","z":"a2a9f2ae6b048ed5","name":"spacer","group":"5652fd7ca1ffc7cb","order":11,"width":6,"height":1},{"id":"8a13ae29a84e697f","type":"ui_group","name":"ΔΙΑΣΤΗΜΑ
ΜΕΤΡΗΣΕΩΝ","tab":"514435e2b139ad59","order":8,"disp":true,"width":"6","collapse":false},{"id":"8ddaa6c080d6a640","type":"ui_group","name":"ΧΕΙΡΙΣΜΟΣ
ΛΕΒΗΤΑ","tab":"514435e2b139ad59","order":9,"disp":true,"width":"6","collapse":false},{"id":"161ec0b4c18e65d4","type":"ui_group","name":"ΗΛΙΑΚΗ
ΑΚΤΙΝΟΒΟΛΙΑ","tab":"514435e2b139ad59","order":6,"disp":true,"width":"6","collapse":false},{"id":"62fba60e072d34be","type":"ui_group","name":"ΕΣΩΤΕΡΙΚΗ
ΘΕΡΜΟΚΡΑΣΙΑ","tab":"6eff3ecd4e79cb10","order":1,"disp":true,"width":"12","collapse":false},{"id":"64404fe9dc1bca5a","type":"ui_group","name":"ΕΣΩΤΕΡΙΚΗ
ΥΓΡΑΣΙΑ","tab":"6eff3ecd4e79cb10","order":2,"disp":true,"width":"12","collapse":false},{"id":"3994db01bbb18560","type":"ui_group","name":"ΘΕΡΜΟΚΡΑΣΙΑ
ΣΩΜΑΤΟΣ","tab":"6eff3ecd4e79cb10","order":3,"disp":true,"width":"12","collapse":false},{"id":"f8b38a7c12d59899","type":"ui_group","name":"ΘΕΡΜΟΚΡΑΣΙΑ
ΣΩΛΗΝΑ","tab":"6eff3ecd4e79cb10","order":4,"disp":true,"width":"12","collapse":false},{"id":"21e992e7178d80bb","type":"ui_group","name":"ΕΞΩΤΕΡΙΚΗ
ΘΕΡΜΟΚΡΑΣΙΑ","tab":"905ff487abaf2841","order":1,"disp":true,"width":"12","collapse":false},{"id":"34ba49d1be34d456","type":"ui_group","name":"ΕΞΩΤΕΡΙΚΗ
ΥΓΡΑΣΙΑ","tab":"905ff487abaf2841","order":2,"disp":true,"width":"12","collapse":false},{"id":"119f14bee3013872","type":"ui_group","name":"ΗΛΙΑΚΗ
ΑΚΤΙΝΟΒΟΛΙΑ","tab":"905ff487abaf2841","order":3,"disp":true,"width":"12","collapse":false},{"id":"c99622170848b903","type":"ui_group","name":"ΘΕΡΜΟΚΡΑΣΙΑ
ΧΩΡΟΥ","tab":"82b494d49f26b61c","order":1,"disp":true,"width":"12","collapse":false},{"id":"eab6edb807d68f4c","type":"ui_group","name":"ΥΓΡΑΣΙΑ
ΧΩΡΟΥ","tab":"82b494d49f26b61c","order":2,"disp":true,"width":"12","collapse":false},{"id":"4b81c4bad649c644","type":"ui_group","name":"ΘΕΡΜΟΚΡΑΣΙΑ
ΣΩΛΗΝΑ","tab":"82b494d49f26b61c","order":4,"disp":true,"width":"12","collapse":false},{"id":"cc0283bfd3c1bee9","type":"ui_tab","name":"Tab

5","icon":"dashboard","order":5},{"id":"16d3494b656538f0","type":"ui_group","name":"
ΘΕΡΜΟΚΡΑΣΙΑ
SERVER","tab":"514435e2b139ad59","order":9,"disp":true,"width":"6","collapse":false},
{"id":"3681e0197122aad9","type":"mqtt
in","z":"a2a9f2ae6b048ed5","name":"","topic":"home/node1/temp","qos":"2","datatype":"
auto","broker":"215adc636c610a74","nl":false,"rap":true,"rh":0,"x":200,"y":100,"wires":[[
"335e3efded8fbe50","deee03e5931f11b3","b73340d140592f6a","3ba554a9f2f28425","afd
43606c037c1f6"]]},{"id":"335e3efded8fbe50","type":"debug","z":"a2a9f2ae6b048ed5","n
ame":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"false","s
tatusVal":"","statusType":"auto","x":440,"y":40,"wires":[]},{"id":"deee03e5931f11b3","ty
pe":"ui_gauge","z":"a2a9f2ae6b048ed5","name":"","group":"a8b2374324598934","order"
:1,"width":6,"height":4,"gtype":"gage","title":"Node1Temp","label":"°C","format":"{{val
ue}}","min":"-
10","max":"60","colors":["#00b500","#00b500","#ca3838"],"seg1":"","seg2":"","x":450,"
y":100,"wires":[]},{"id":"b73340d140592f6a","type":"function","z":"a2a9f2ae6b048ed5",
"name":"convertToNumber","func":"msg.payload = Number(msg.payload);\nreturn
msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":470,"y":160,"wires":[[
"c43c09dbdd95dadf"]]},{"id":"c43c09dbdd95dadf","type":"influxdb
out","z":"a2a9f2ae6b048ed5","influxdb":"efcb09695f40f060","name":"node1Temp","mea
surement":"node1Temp","precision":"","retentionPolicy":"","database":"database","precisi
onV18FluxV20":"ms","retentionPolicyV18Flux":"","org":"organisation","bucket":"bucket
","x":740,"y":160,"wires":[]},{"id":"60ceae2cf6d8bdaa","type":"mqtt
in","z":"a2a9f2ae6b048ed5","name":"","topic":"home/node1/hum","qos":"2","datatype":"
auto","broker":"215adc636c610a74","nl":false,"rap":true,"rh":0,"x":220,"y":420,"wires":[[
"9616b46a9638a779","752432df41582c88","390370c67b60a7b9","827ebf297bc7389e","3
a1aa9323cbabdf5"]]},{"id":"9616b46a9638a779","type":"debug","z":"a2a9f2ae6b048ed5"
,"name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"false
","statusVal":"","statusType":"auto","x":450,"y":360,"wires":[]},{"id":"752432df41582c8
8","type":"ui_gauge","z":"a2a9f2ae6b048ed5","name":"","group":"42eff3bb8dea9da5","or
der":1,"width":6,"height":4,"gtype":"gage","title":"Node1Hum","label":"%","format":"{{v
alue}}","min":"0","max":"100","colors":["#00b500","#00b500","#ca3838"],"seg1":"","seg
2":"","x":460,"y":420,"wires":[]},{"id":"390370c67b60a7b9","type":"function","z":"a2a9f
2ae6b048ed5","name":"convertToNumber","func":"msg.payload =
Number(msg.payload);\nreturn
msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":480,"y":480,"wires":[[
"ac39e3a147ccb8a0"]]},{"id":"ac39e3a147ccb8a0","type":"influxdb
out","z":"a2a9f2ae6b048ed5","influxdb":"efcb09695f40f060","name":"node1Hum","meas
urement":"node1Hum","precision":"","retentionPolicy":"","database":"database","precisio
nV18FluxV20":"ms","retentionPolicyV18Flux":"","org":"organisation","bucket":"bucket"
,"x":750,"y":480,"wires":[]},{"id":"9d1b17e173c2da55","type":"mqtt
in","z":"a2a9f2ae6b048ed5","name":"","topic":"home/node2/temp","qos":"2","datatype":"
auto","broker":"215adc636c610a74","nl":false,"rap":true,"rh":0,"x":230,"y":740,"wires":[[
"04e0436bf061e90f","923349c3923ce2a7","ec5eaea378549b78","eff4b55a4791d2bf","af4
727f486bef395"]]},{"id":"04e0436bf061e90f","type":"debug","z":"a2a9f2ae6b048ed5","n
ame":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"false","s
tatusVal":"","statusType":"auto","x":470,"y":680,"wires":[]},{"id":"923349c3923ce2a7","
type":"ui_gauge","z":"a2a9f2ae6b048ed5","name":"","group":"27ceab0bc7228cc3","order
":1,"width":6,"height":4,"gtype":"gage","title":"Node2Temp","label":"°C","format":"{{val
ue}}","min":"-

55","max":"125","colors":["#ff4d00","#ff4d00","#ca3838"],"seg1":"","seg2":"","x":480,"y
":740,"wires":[]},{"id":"ec5eaea378549b78","type":"function","z":"a2a9f2ae6b048ed5","
name":"convertToNumber","func":"msg.payload = Number(msg.payload);\nreturn
msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":500,"y":800,"wires":[[
"c3d7e083b2f916bb"]]},{"id":"c3d7e083b2f916bb","type":"influxdb
out","z":"a2a9f2ae6b048ed5","influxdb":"efcb09695f40f060","name":"node2Temp","mea
surement":"node2Temp","precision":"","retentionPolicy":"","database":"database","precisi
onV18FluxV20":"ms","retentionPolicyV18Flux":"","org":"organisation","bucket":"bucket
","x":770,"y":800,"wires":[]},{"id":"a25b69a4e98481c4","type":"mqtt
in","z":"a2a9f2ae6b048ed5","name":"","topic":"home/node3/temp","qos":"2","datatype":"
auto","broker":"215adc636c610a74","nl":false,"rap":true,"rh":0,"x":250,"y":1060,"wires":
[["ab3d7c6274afe69c","3cc91edd81ca25d3","2493e8dcf6fd0770","cc3770c366f2cd87"]]}
,{"id":"ab3d7c6274afe69c","type":"debug","z":"a2a9f2ae6b048ed5","name":"","active":tr
ue,"tosidebar":true,"console":false,"tostatus":false,"complete":"false","statusVal":"","statu
sType":"auto","x":490,"y":1000,"wires":[]},{"id":"3cc91edd81ca25d3","type":"ui_gauge"
,"z":"a2a9f2ae6b048ed5","name":"","group":"f8b38a7c12d59899","order":1,"width":6,"he
ight":4,"gtype":"gage","title":"Node3Temp","label":"°C","format":"{{value}}","min":"-
55","max":"125","colors":["#00e1ff","#00ffff","#ca3838"],"seg1":"","seg2":"","x":500,"y"
:1060,"wires":[]},{"id":"2493e8dcf6fd0770","type":"function","z":"a2a9f2ae6b048ed5","
name":"convertToNumber","func":"msg.payload = Number(msg.payload);\nreturn
msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":520,"y":1120,"wires":[
["f150f1d30ba37a0d"]]},{"id":"f150f1d30ba37a0d","type":"influxdb
out","z":"a2a9f2ae6b048ed5","influxdb":"efcb09695f40f060","name":"node3Temp","mea
surement":"node3Temp","precision":"","retentionPolicy":"","database":"database","precisi
onV18FluxV20":"ms","retentionPolicyV18Flux":"","org":"organisation","bucket":"bucket
","x":790,"y":1120,"wires":[]},{"id":"18e5a91f118a13cd","type":"mqtt
in","z":"a2a9f2ae6b048ed5","name":"","topic":"home/node4/temp","qos":"2","datatype":"
auto","broker":"215adc636c610a74","nl":false,"rap":true,"rh":0,"x":250,"y":1340,"wires":
[["ac6220329752af7a","afc0d4b531f5775b","367762cf6c1d0cb6","7a25c51994aa96e7"]]}
,{"id":"ac6220329752af7a","type":"debug","z":"a2a9f2ae6b048ed5","name":"","active":tr
ue,"tosidebar":true,"console":false,"tostatus":false,"complete":"false","statusVal":"","statu
sType":"auto","x":490,"y":1280,"wires":[]},{"id":"afc0d4b531f5775b","type":"ui_gauge",
"z":"a2a9f2ae6b048ed5","name":"","group":"62fba60e072d34be","order":3,"width":6,"hei
ght":4,"gtype":"gage","title":"Node4Temp","label":"°C","format":"{{value}}","min":"-
10","max":"60","colors":["#5000b4","#5000b4","#ca3838"],"seg1":"","seg2":"","x":500,"
y":1340,"wires":[]},{"id":"367762cf6c1d0cb6","type":"function","z":"a2a9f2ae6b048ed5"
,"name":"convertToNumber","func":"msg.payload = Number(msg.payload);\nreturn
msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":510,"y":1400,"wires":[
["46cc7704d4fcbb5f"]]},{"id":"46cc7704d4fcbb5f","type":"influxdb
out","z":"a2a9f2ae6b048ed5","influxdb":"efcb09695f40f060","name":"node4Temp","mea
surement":"node4Temp","precision":"","retentionPolicy":"","database":"database","precisi
onV18FluxV20":"ms","retentionPolicyV18Flux":"","org":"organisation","bucket":"bucket
","x":790,"y":1400,"wires":[]},{"id":"b0535319571ea061","type":"mqtt
in","z":"a2a9f2ae6b048ed5","name":"","topic":"home/node4/hum","qos":"2","datatype":"
auto","broker":"215adc636c610a74","nl":false,"rap":true,"rh":0,"x":240,"y":1600,"wires":
[["95cb3ac394799a4c","3e721fc9cd7be0cc","ce93859c0dd49acb","939bd5b221226b97"]]
},{"id":"95cb3ac394799a4c","type":"debug","z":"a2a9f2ae6b048ed5","name":"","active":
true,"tosidebar":true,"console":false,"tostatus":false,"complete":"false","statusVal":"","stat
usType":"auto","x":490,"y":1540,"wires":[]},{"id":"3e721fc9cd7be0cc","type":"ui_gauge

","z":"a2a9f2ae6b048ed5","name":"","group":"64404fe9dc1bca5a","order":3,"width":6,"h
eight":4,"gtype":"gage","title":"Node4Hum","label":"%","format":"{{value}}","min":"0",
"max":"100","colors":["#5000b4","#5000b4","#ca3838"],"seg1":"","seg2":"","x":500,"y":
1600,"wires":[]},{"id":"ce93859c0dd49acb","type":"function","z":"a2a9f2ae6b048ed5","n
ame":"convertToNumber","func":"msg.payload = Number(msg.payload);\nreturn
msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":520,"y":1660,"wires":[
["d9d881cf95ab4bc1"]]},{"id":"d9d881cf95ab4bc1","type":"influxdb
out","z":"a2a9f2ae6b048ed5","influxdb":"efcb09695f40f060","name":"node4Hum","meas
urement":"node4Hum","precision":"","retentionPolicy":"","database":"database","precisio
nV18FluxV20":"ms","retentionPolicyV18Flux":"","org":"organisation","bucket":"bucket"
,"x":790,"y":1660,"wires":[]},{"id":"1029a07242629bf2","type":"mqtt
out","z":"a5b7d3d6cf06605c","name":"","topic":"home/time","qos":"1","retain":"true","re
spTopic":"","contentType":"","userProps":"","correl":"","expiry":"","broker":"215adc636c
610a74","x":410,"y":100,"wires":[]},{"id":"0ef252f43134f258","type":"ui_dropdown","z"
:"a5b7d3d6cf06605c","name":"","label":"","tooltip":"","place":"Select
option","group":"8a13ae29a84e697f","order":1,"width":6,"height":2,"passthru":true,"mult
iple":false,"options":[{"label":"-","value":0,"type":"num"},{"label":"1
λεπτό","value":1,"type":"num"},{"label":"2 λεπτά","value":2,"type":"num"},{"label":"3
λεπτά","value":3,"type":"num"},{"label":"4 λεπτά","value":4,"type":"num"},{"label":"5
λεπτά","value":5,"type":"num"},{"label":"6 λεπτά","value":6,"type":"num"},{"label":"7
λεπτά","value":7,"type":"num"},{"label":"8 λεπτά","value":8,"type":"num"},{"label":"9
λεπτά","value":9,"type":"num"},{"label":"10
λεπτά","value":10,"type":"num"},{"label":"11
λεπτά","value":11,"type":"num"},{"label":"12
λεπτά","value":12,"type":"num"},{"label":"13
λεπτά","value":13,"type":"num"},{"label":"14
λεπτά","value":14,"type":"num"},{"label":"15
λεπτά","value":15,"type":"num"}],"payload":"","topic":"topic","topicType":"msg","x":16
0,"y":100,"wires":[["dc2cbc6b980039f7","1029a07242629bf2"]]},{"id":"dc2cbc6b980039
f7","type":"debug","z":"a5b7d3d6cf06605c","name":"","active":true,"tosidebar":true,"con
sole":false,"tostatus":false,"complete":"false","statusVal":"","statusType":"auto","x":410,"
y":40,"wires":[]},{"id":"15be3bf0b1476b1b","type":"mqtt
in","z":"b1ab34a6f7efaa74","name":"","topic":"home/node5/temp","qos":"2","datatype":"
auto","broker":"215adc636c610a74","nl":false,"rap":true,"rh":0,"x":170,"y":100,"wires":[[
"b608d218c45a7ade","c21d07f2bbc5d3a5","4c6f18645c57c9a0","4577c5b3427a2e98","0f
023b5fdcd5fc2e"]]},{"id":"b608d218c45a7ade","type":"debug","z":"b1ab34a6f7efaa74","
name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"false",
"statusVal":"","statusType":"auto","x":410,"y":40,"wires":[]},{"id":"4c6f18645c57c9a0","
type":"ui_gauge","z":"b1ab34a6f7efaa74","name":"","group":"4724563832291ce5","order
":1,"width":6,"height":4,"gtype":"gage","title":"Node5Temp","label":"°C","format":"{{val
ue}}","min":"-
10","max":"60","colors":["#0011ff","#0011ff","#ca3838"],"seg1":"","seg2":"","x":420,"y"
:100,"wires":[]},{"id":"c21d07f2bbc5d3a5","type":"function","z":"b1ab34a6f7efaa74","na
me":"convertToNumber","func":"msg.payload = Number(msg.payload);\nreturn
msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":440,"y":160,"wires":[[
"16b568a00f662dfe"]]},{"id":"16b568a00f662dfe","type":"influxdb
out","z":"b1ab34a6f7efaa74","influxdb":"efcb09695f40f060","name":"node5Temp","mea
surement":"node5Temp","precision":"","retentionPolicy":"","database":"database","precisi
onV18FluxV20":"ms","retentionPolicyV18Flux":"","org":"organisation","bucket":"bucket

","x":710,"y":160,"wires":[]},{"id":"4b160eeda5a75600","type":"mqtt
in","z":"b1ab34a6f7efaa74","name":"","topic":"home/node5/hum","qos":"2","datatype":"a
uto","broker":"215adc636c610a74","nl":false,"rap":true,"rh":0,"x":160,"y":420,"wires":[["
8ee85dd80d37b28c","e3809951170a4440","ab856f20d07652c9","0bb1242c5180082f","9
dfadacc835bea82"]]},{"id":"8ee85dd80d37b28c","type":"debug","z":"b1ab34a6f7efaa74",
"name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"false"
,"statusVal":"","statusType":"auto","x":410,"y":360,"wires":[]},{"id":"ab856f20d07652c9
","type":"ui_gauge","z":"b1ab34a6f7efaa74","name":"","group":"969f4575dbd73cb8","or
der":1,"width":6,"height":4,"gtype":"gage","title":"Node5Hum","label":"%","format":"{{v
alue}}","min":"0","max":"100","colors":["#0011ff","#0011ff","#ca3838"],"seg1":"","seg2
":"","x":420,"y":420,"wires":[]},{"id":"e3809951170a4440","type":"function","z":"b1ab3
4a6f7efaa74","name":"convertToNumber","func":"msg.payload =
Number(msg.payload);\nreturn
msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":440,"y":480,"wires":[[
"8ff83c680f667291"]]},{"id":"8ff83c680f667291","type":"influxdb
out","z":"b1ab34a6f7efaa74","influxdb":"efcb09695f40f060","name":"node5Hum","meas
urement":"node5Hum","precision":"","retentionPolicy":"","database":"database","precisio
nV18FluxV20":"ms","retentionPolicyV18Flux":"","org":"organisation","bucket":"bucket"
,"x":710,"y":480,"wires":[]},{"id":"5245b6178e224c2e","type":"mqtt
in","z":"1e1d5d4fe0c4f0d6","name":"","topic":"home/node6/tempDHT","qos":"2","dataty
pe":"auto","broker":"215adc636c610a74","nl":false,"rap":true,"rh":0,"x":160,"y":100,"wir
es":[["6291edfe15e532ed","c625b3e979591c03","3c3366f8fa806ea9","3096cb35d830fef0
"]]},{"id":"6291edfe15e532ed","type":"debug","z":"1e1d5d4fe0c4f0d6","name":"","active
":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"false","statusVal":"","st
atusType":"auto","x":390,"y":40,"wires":[]},{"id":"c625b3e979591c03","type":"ui_gauge
","z":"1e1d5d4fe0c4f0d6","name":"","group":"c99622170848b903","order":1,"width":6,"
height":4,"gtype":"gage","title":"Node6TempDHT","label":"°C","format":"{{value}}","m
in":"-
10","max":"60","colors":["#ff0000","#ff0000","#ca3838"],"seg1":"","seg2":"","x":420,"y"
:100,"wires":[]},{"id":"3c3366f8fa806ea9","type":"function","z":"1e1d5d4fe0c4f0d6","na
me":"convertToNumber","func":"msg.payload = Number(msg.payload);\nreturn
msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":420,"y":160,"wires":[[
"83a72de69dabff38"]]},{"id":"83a72de69dabff38","type":"influxdb
out","z":"1e1d5d4fe0c4f0d6","influxdb":"efcb09695f40f060","name":"node6TempDHT",
"measurement":"node6TempDHT","precision":"","retentionPolicy":"","database":"databas
e","precisionV18FluxV20":"ms","retentionPolicyV18Flux":"","org":"organisation","buck
et":"bucket","x":700,"y":160,"wires":[]},{"id":"5c376cfb3a41b8bf","type":"mqtt
in","z":"1e1d5d4fe0c4f0d6","name":"","topic":"home/node6/humDHT","qos":"2","dataty
pe":"auto","broker":"215adc636c610a74","nl":false,"rap":true,"rh":0,"x":160,"y":360,"wir
es":[["9393198594106e8b","fa5a2eb1d0950fc6","cc15da42faadc6af","935776058a348ee9
"]]},{"id":"9393198594106e8b","type":"debug","z":"1e1d5d4fe0c4f0d6","name":"","activ
e":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"false","statusVal":"","
statusType":"auto","x":390,"y":300,"wires":[]},{"id":"fa5a2eb1d0950fc6","type":"ui_gau
ge","z":"1e1d5d4fe0c4f0d6","name":"","group":"eab6edb807d68f4c","order":1,"width":6,
"height":4,"gtype":"gage","title":"Node6HumDHT","label":"%","format":"{{value}}","mi
n":"0","max":"100","colors":["#ff0000","#ff0000","#ca3838"],"seg1":"","seg2":"","x":410
,"y":360,"wires":[]},{"id":"cc15da42faadc6af","type":"function","z":"1e1d5d4fe0c4f0d6",
"name":"convertToNumber","func":"msg.payload = Number(msg.payload);\nreturn
msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":420,"y":420,"wires":[[

"38790f940f11164b"]]},{"id":"38790f940f11164b","type":"influxdb
out","z":"1e1d5d4fe0c4f0d6","influxdb":"efcb09695f40f060","name":"node6HumDHT","
measurement":"node6HumDHT","precision":"","retentionPolicy":"","database":"database
","precisionV18FluxV20":"ms","retentionPolicyV18Flux":"","org":"organisation","bucket
":"bucket","x":700,"y":420,"wires":[]},{"id":"913fb495419da126","type":"debug","z":"1e
1d5d4fe0c4f0d6","name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"
complete":"false","statusVal":"","statusType":"auto","x":430,"y":800,"wires":[]},{"id":"f8
ad5055332cd5d9","type":"mqtt
out","z":"1e1d5d4fe0c4f0d6","name":"","topic":"home/node6/relay","qos":"1","retain":"tr
ue","respTopic":"","contentType":"","userProps":"","correl":"","expiry":"","broker":"215a
dc636c610a74","x":450,"y":880,"wires":[]},{"id":"a1e7abdaba972f4f","type":"ui_switch",
"z":"1e1d5d4fe0c4f0d6","name":"Relay","label":"Άνοιγμα
Θέρμανσης","tooltip":"","group":"8ddaa6c080d6a640","order":1,"width":6,"height":2,"pa
ssthru":true,"decouple":"false","topic":"","topicType":"str","style":"","onvalue":"ON","on
valueType":"str","onicon":"","oncolor":"","offvalue":"OFF","offvalueType":"str","officon
":"","offcolor":"","animate":false,"x":190,"y":880,"wires":[["913fb495419da126","f8ad50
55332cd5d9"]]},{"id":"b23c4fa793665838","type":"mqtt
in","z":"1e1d5d4fe0c4f0d6","name":"","topic":"home/node6/tempDS","qos":"2","datatype
":"auto","broker":"215adc636c610a74","nl":false,"rap":true,"rh":0,"x":190,"y":620,"wires
":[["bb6775d058bc8a3c","4d27970c41bb3ceb","fa55659f1de1912c","05d43b779dda8480
"]]},{"id":"bb6775d058bc8a3c","type":"debug","z":"1e1d5d4fe0c4f0d6","name":"","activ
e":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"false","statusVal":"","
statusType":"auto","x":420,"y":560,"wires":[]},{"id":"4d27970c41bb3ceb","type":"ui_gau
ge","z":"1e1d5d4fe0c4f0d6","name":"","group":"4b81c4bad649c644","order":1,"width":6
,"height":4,"gtype":"gage","title":"Node6TempDS","label":"°C","format":"{{value}}","mi
n":"-
55","max":"125","colors":["#ff00ff","#ff00ff","#ca3838"],"seg1":"","seg2":"","x":440,"y":
620,"wires":[]},{"id":"fa55659f1de1912c","type":"function","z":"1e1d5d4fe0c4f0d6","na
me":"convertToNumber","func":"msg.payload = Number(msg.payload);\nreturn
msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":450,"y":680,"wires":[[
"130f1c8d69881fbe"]]},{"id":"130f1c8d69881fbe","type":"influxdb
out","z":"1e1d5d4fe0c4f0d6","influxdb":"efcb09695f40f060","name":"node6TempDS","
measurement":"node6TempDS","precision":"","retentionPolicy":"","database":"database",
"precisionV18FluxV20":"ms","retentionPolicyV18Flux":"","org":"organisation","bucket":
"bucket","x":730,"y":680,"wires":[]},{"id":"0b104766535c12aa","type":"ui_gauge","z":"b
1ab34a6f7efaa74","name":"","group":"161ec0b4c18e65d4","order":1,"width":6,"height":4
,"gtype":"gage","title":"Node5Sun","label":"W/m2","format":"{{value}}","min":"0","max
":"820","colors":["#e6e600","#e6e600","#ca3838"],"seg1":"","seg2":"","x":430,"y":740,"
wires":[]},{"id":"3ba554a9f2f28425","type":"ui_gauge","z":"a2a9f2ae6b048ed5","name":
"","group":"62fba60e072d34be","order":1,"width":6,"height":4,"gtype":"gage","title":"No
de1Temp","label":"°C","format":"{{value}}","min":"-
10","max":"60","colors":["#00b500","#00b500","#ca3838"],"seg1":"","seg2":"","x":450,"
y":220,"wires":[]},{"id":"827ebf297bc7389e","type":"ui_gauge","z":"a2a9f2ae6b048ed5",
"name":"","group":"64404fe9dc1bca5a","order":1,"width":6,"height":4,"gtype":"gage","tit
le":"Node1Hum","label":"%","format":"{{value}}","min":"0","max":"100","colors":["#00
b500","#00b500","#ca3838"],"seg1":"","seg2":"","x":470,"y":540,"wires":[]},{"id":"afd43
606c037c1f6","type":"ui_chart","z":"a2a9f2ae6b048ed5","name":"","group":"62fba60e07
2d34be","order":2,"width":6,"height":4,"label":"Node1Temp","chartType":"line","legend"
:"false","xformat":"HH:mm:ss","interpolate":"linear","nodata":"","dot":true,"ymin":"","y

max":"","removeOlder":1,"removeOlderPoints":"","removeOlderUnit":"3600","cutout":0,
"useOneColor":false,"useUTC":false,"colors":["#00b500","#aec7e8","#ff7f0e","#0077b3"
,"#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"outputs":1,"useDifferentColor":f
alse,"x":450,"y":280,"wires":[[]]},{"id":"3a1aa9323cbabdf5","type":"ui_chart","z":"a2a9f
2ae6b048ed5","name":"","group":"64404fe9dc1bca5a","order":2,"width":6,"height":4,"lab
el":"Node1Hum","chartType":"line","legend":"false","xformat":"HH:mm:ss","interpolate"
:"linear","nodata":"","dot":true,"ymin":"","ymax":"","removeOlder":1,"removeOlderPoint
s":"","removeOlderUnit":"3600","cutout":0,"useOneColor":false,"useUTC":false,"colors":
["#00b500","#aec7e8","#ff7f0e","#2c64a0","#98df8a","#d62728","#ff9896","#9467bd","#
c5b0d5"],"outputs":1,"useDifferentColor":false,"x":470,"y":600,"wires":[[]]},{"id":"eff4b
55a4791d2bf","type":"ui_gauge","z":"a2a9f2ae6b048ed5","name":"","group":"3994db01b
bb18560","order":1,"width":6,"height":4,"gtype":"gage","title":"Node2Temp","label":"°C
","format":"{{value}}","min":"-
55","max":"125","colors":["#ff4d00","#ff4d00","#ca3838"],"seg1":"","seg2":"","x":490,"y
":860,"wires":[]},{"id":"af4727f486bef395","type":"ui_chart","z":"a2a9f2ae6b048ed5","n
ame":"","group":"3994db01bbb18560","order":2,"width":6,"height":4,"label":"Node2Tem
p","chartType":"line","legend":"false","xformat":"HH:mm:ss","interpolate":"linear","noda
ta":"","dot":true,"ymin":"","ymax":"","removeOlder":1,"removeOlderPoints":"","remove
OlderUnit":"3600","cutout":0,"useOneColor":false,"useUTC":false,"colors":["#ff4d00","#
aec7e8","#ff7f0e","#2ca02c","#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"out
puts":1,"useDifferentColor":false,"x":500,"y":920,"wires":[[]]},{"id":"cc3770c366f2cd87"
,"type":"ui_chart","z":"a2a9f2ae6b048ed5","name":"","group":"f8b38a7c12d59899","orde
r":2,"width":6,"height":4,"label":"Node3Temp","chartType":"line","legend":"false","xfor
mat":"HH:mm:ss","interpolate":"linear","nodata":"","dot":true,"ymin":"","ymax":"","remo
veOlder":1,"removeOlderPoints":"","removeOlderUnit":"3600","cutout":0,"useOneColor"
:false,"useUTC":false,"colors":["#00ffff","#aec7e8","#ff7f0e","#2ca02c","#98df8a","#d62
728","#ff9896","#9467bd","#c5b0d5"],"outputs":1,"useDifferentColor":false,"x":510,"y":
1180,"wires":[[]]},{"id":"7a25c51994aa96e7","type":"ui_chart","z":"a2a9f2ae6b048ed5",
"name":"","group":"62fba60e072d34be","order":4,"width":6,"height":4,"label":"Node4Te
mp","chartType":"line","legend":"false","xformat":"HH:mm:ss","interpolate":"linear","no
data":"","dot":true,"ymin":"","ymax":"","removeOlder":1,"removeOlderPoints":"","remov
eOlderUnit":"3600","cutout":0,"useOneColor":false,"useUTC":false,"colors":["#5000b4",
"#aec7e8","#ff7f0e","#2ca02c","#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"o
utputs":1,"useDifferentColor":false,"x":490,"y":1460,"wires":[[]]},{"id":"939bd5b221226
b97","type":"ui_chart","z":"a2a9f2ae6b048ed5","name":"","group":"64404fe9dc1bca5a","
order":4,"width":6,"height":4,"label":"Node4Hum","chartType":"line","legend":"false","x
format":"HH:mm:ss","interpolate":"linear","nodata":"","dot":true,"ymin":"","ymax":"","re
moveOlder":1,"removeOlderPoints":"","removeOlderUnit":"3600","cutout":0,"useOneCol
or":false,"useUTC":false,"colors":["#5000b4","#aec7e8","#ff7f0e","#2ca02c","#98df8a","
#d62728","#ff9896","#9467bd","#c5b0d5"],"outputs":1,"useDifferentColor":false,"x":510
,"y":1720,"wires":[[]]},{"id":"4577c5b3427a2e98","type":"ui_gauge","z":"b1ab34a6f7efa
a74","name":"","group":"21e992e7178d80bb","order":1,"width":6,"height":4,"gtype":"gag
e","title":"Node5Temp","label":"°C","format":"{{value}}","min":"-
10","max":"60","colors":["#0011ff","#0011ff","#ca3838"],"seg1":"","seg2":"","x":430,"y"
:220,"wires":[]},{"id":"0f023b5fdcd5fc2e","type":"ui_chart","z":"b1ab34a6f7efaa74","na
me":"","group":"21e992e7178d80bb","order":2,"width":6,"height":4,"label":"Node5Temp
","chartType":"line","legend":"false","xformat":"HH:mm:ss","interpolate":"linear","nodat
a":"","dot":true,"ymin":"-
","ymax":"","removeOlder":1,"removeOlderPoints":"","removeOlderUnit":"3600","cutout

":0,"useOneColor":false,"useUTC":false,"colors":["#0011ff","#aec7e8","#ff7f0e","#2ca02c","#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"outputs":1,"useDifferentColor":false,"x":430,"y":280,"wires":[[]]},{"id":"0bb1242c5180082f","type":"ui_gauge","z":"b1ab34a6f7efaa74","name":"","group":"34ba49d1be34d456","order":1,"width":6,"height":4,"gtype":"gage","title":"Node5Hum","label":"%","format":"{{value}}","min":"0","max":"100","colors":["#0011ff","#0011ff","#ca3838"],"seg1":"","seg2":"","x":430,"y":540,"wires":[]},{"id":"9dfadacc835bea82","type":"ui_chart","z":"b1ab34a6f7efaa74","name":"","group":"34ba49d1be34d456","order":2,"width":6,"height":4,"label":"Node5Hum","chartType":"line","legend":"false","xformat":"HH:mm:ss","interpolate":"linear","nodata":"","dot":true,"ymin":"","ymax":"","removeOlder":1,"removeOlderPoints":"","removeOlderUnit":"3600","cutout":0,"useOneColor":false,"useUTC":false,"colors":["#0011ff","#aec7e8","#ff7f0e","#2ca02c","#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"outputs":1,"useDifferentColor":false,"x":430,"y":600,"wires":[[]]},{"id":"3421cffa699b9090","type":"ui_gauge","z":"b1ab34a6f7efaa74","name":"","group":"119f14bee3013872","order":1,"width":6,"height":4,"gtype":"gage","title":"Node5Sun","label":"W/m2","format":"{{value}}","min":"0","max":"820","colors":["#e6e600","#e6e600","#ca3838"],"seg1":"","seg2":"","x":430,"y":860,"wires":[]},{"id":"90250756262c278c","type":"ui_chart","z":"b1ab34a6f7efaa74","name":"","group":"119f14bee3013872","order":2,"width":6,"height":4,"label":"Node5Sun","chartType":"line","legend":"false","xformat":"HH:mm:ss","interpolate":"linear","nodata":"","dot":true,"ymin":"","ymax":"","removeOlder":1,"removeOlderPoints":"","removeOlderUnit":"3600","cutout":0,"useOneColor":false,"useUTC":false,"colors":["#e6e600","#aec7e8","#ff7f0e","#2ca02c","#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"outputs":1,"useDifferentColor":false,"x":430,"y":920,"wires":[[]]},{"id":"3096cb35d830fef0","type":"ui_chart","z":"1e1d5d4fe0c4f0d6","name":"","group":"c99622170848b903","order":2,"width":6,"height":4,"label":"Node6TempDHT","chartType":"line","legend":"false","xformat":"HH:mm:ss","interpolate":"linear","nodata":"","dot":true,"ymin":"","ymax":"","removeOlder":1,"removeOlderPoints":"","removeOlderUnit":"3600","cutout":0,"useOneColor":false,"useUTC":false,"colors":["#ff0000","#aec7e8","#ff7f0e","#2ca02c","#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"outputs":1,"useDifferentColor":false,"x":430,"y":220,"wires":[[]]},{"id":"05d43b779dda8480","type":"ui_chart","z":"1e1d5d4fe0c4f0d6","name":"","group":"4b81c4bad649c644","order":2,"width":6,"height":4,"label":"Node6TempDS","chartType":"line","legend":"false","xformat":"HH:mm:ss","interpolate":"linear","nodata":"","dot":true,"ymin":"","ymax":"","removeOlder":1,"removeOlderPoints":"","removeOlderUnit":"3600","cutout":0,"useOneColor":false,"useUTC":false,"colors":["#ff00ff","#aec7e8","#ff7f0e","#2ca02c","#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"outputs":1,"useDifferentColor":false,"x":440,"y":740,"wires":[[]]},{"id":"935776058a348ee9","type":"ui_chart","z":"1e1d5d4fe0c4f0d6","name":"","group":"eab6edb807d68f4c","order":2,"width":6,"height":4,"label":"Node6HumDHT","chartType":"line","legend":"false","xformat":"HH:mm:ss","interpolate":"linear","nodata":"","dot":true,"ymin":"","ymax":"","removeOlder":1,"removeOlderPoints":"","removeOlderUnit":"3600","cutout":0,"useOneColor":false,"useUTC":false,"colors":["#ff0000","#aec7e8","#ff7f0e","#2ca02c","#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"outputs":1,"useDifferentColor":false,"x":420,"y":480,"wires":[[]]},{"id":"f7052b5ffe3c824b","type":"mqtt in","z":"b1ab34a6f7efaa74","name":"","topic":"home/node5/sun","qos":"2","datatype":"auto","broker":"215adc636c610a74","nl":false,"rap":true,"rh":0,"x":180,"y":740,"wires":[["81d28d10a9a435f6","0b104766535c12aa","054fa434f7d293a6","3421cffa699b9090","90250756262c278c"]]},{"id":"81d28d10a9a435f6","type":"debug","z":"b1ab34a6f7efaa74","name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"false","statusVal":"","statusType":"auto","x":430,"y":680,"wires":[]},{"id":"054fa434f7d293a6",

"type":"function","z":"b1ab34a6f7efaa74","name":"convertToNumber","func":"msg.payload = Number(msg.payload);\nreturn msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":450,"y":800,"wires":[["3359e644dcd529b4"]]},{"id":"3359e644dcd529b4","type":"influxdb out","z":"b1ab34a6f7efaa74","influxdb":"efcb09695f40f060","name":"node5Sun","measurement":"node5Sun","precision":"","retentionPolicy":"","database":"database","precisionV18FluxV20":"ms","retentionPolicyV18Flux":"","org":"organisation","bucket":"bucket","x":710,"y":800,"wires":[]},{"id":"19df5e79ca48e9f3","type":"exec","z":"a5b7d3d6cf06605c","command":"vcgencmd measure_temp","addpay":"","append":"","useSpawn":"false","timer":"","winHide":false,"oldrc":false,"name":"","x":410,"y":240,"wires":[["646ebbacb7cd188c","3b7a68053f85f159"],[],[]]},{"id":"646ebbacb7cd188c","type":"debug","z":"a5b7d3d6cf06605c","name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"false","statusVal":"","statusType":"auto","x":650,"y":220,"wires":[]},{"id":"fe87091f4a7fa056","type":"inject","z":"a5b7d3d6cf06605c","name":"","props":[{"p":"payload"},{"p":"topic","vt":"str"}],"repeat":"","crontab":"","once":false,"onceDelay":0.1,"topic":"","payload":"","payloadType":"date","x":160,"y":200,"wires":[["19df5e79ca48e9f3"]]},{"id":"934e7e0149d9bd6b","type":"ui_button","z":"a5b7d3d6cf06605c","name":"","group":"16d3494b656538f0","order":1,"width":"6","height":"1","passthru":false,"label":"button","tooltip":"","color":"","bgcolor":"","icon":"","payload":"","payloadType":"str","topic":"topic","topicType":"msg","x":150,"y":260,"wires":[["19df5e79ca48e9f3"]]},{"id":"3b7a68053f85f159","type":"ui_text","z":"a5b7d3d6cf06605c","group":"16d3494b656538f0","order":2,"width":"6","height":"1","name":"","label":"","format":"{{msg.payload}}","layout":"row-center","x":630,"y":260,"wires":[]}]

Υπεύθυνη Δήλωση Συγγραφέα:

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν.1599/1986, η παρούσα εργασία αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης.