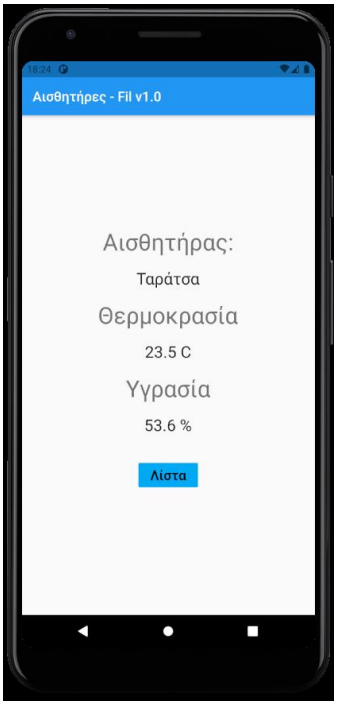
ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Ανάπτυξη android εφαρμογής παρακολούθησης περιβαλλοντικών συνθηκών με raspberry»



Φοιτητής

Φιλοξενίδης Νικόλαος 514321

Επιβλέπων

Δρ. Κυριάκος Τσιακμάκης

Ιούνιος 2021

Ανάπτυξη android εφαρμογής παρακολούθησης περιβαλλοντικών συνθηκών με raspberry

Κωδικός: 20203

Φοιτητής: Φιλοξενίδης Νικόλαος

Εισηγητής: Δρ Κυριάκος Τσιακμάκης

Ημερομηνία ανάληψης Π.Ε. 15-10-2020

Ημερομηνία περάτωσης Π.Ε. 01-06-2021

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως πτυχιακή εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της πτυχιακής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

Περίληψη

Η εργασία αυτή αφορά τη μελέτη και ανάπτυξη android εφαρμογής παρακολούθησης περιβαλλοντικών συνθηκών με raspberry pi 3. Η εφαρμογή android επικοινωνεί με το raspberry μέσω wifi για την παρακολούθηση θερμοκρασίας και υγρασίας σε έναν χώρο. Ο χρήστης μπορεί να ενεργοποιεί την φορητή συσκευή μικρού κόστους και κατανάλωσης και να επιβλέπει με το κινητό του τις περιβαντολλογικές συνθήκες πολύ γρήγορα. Η χρησιμότητα της εφαρμογής είναι η καταμέτρηση της θερμοκρασίας/υγρασίας σε διάφορα σημεία και ύψη ενός σπιτιού ή ενός χώρου. Επίσης, παρέχεται η δυνατότητα προσθήκης περισσότερων κόμβων που μπορεί να διαχειριστεί η εφαρμογή. Τέλος, μια πολύ σημαντική δυνατότητα είναι η επίβλεψη των τιμών των αισθητήρων διαδικτυακά μέσω της υπηρεσίας Thingspeak. Κάθε κόμβος μπορεί να στέλνει τα δεδομένα του μέσω ασφαλούς καναλιού στο Thingspeak και ο χρήστης μέσω της εφαρμογής να τα επιβλέπει από το κινητό του.

« Development of android application for monitoring environmental conditions with raspberry-»

Abstract

This work concerns the study and development of android application for monitoring environmental conditions with raspberry pi. The android application communicates with raspberry via wifi to record temperature and humidity in a home or space. The user can activate the low cost and consumption mobile device and monitor the environmental conditions with his mobile phone very quickly. The usefulness of the application is the measurement of temperature / humidity at various points and heights of a house or a space. It also provides the ability to add more nodes that the application can handle. Finally, an important feature is the monitoring of sensor values via internet with the help of Thingspeak service. Each node can send its data through a secure channel to Thingspeak and the user using the application can monitor the nodes.

Ευχαριστίες

Θέλω να ευχαριστήσω τους γονείς μου για τη πολύτιμη βοήθεια τους.

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Στη τωρινή εποχή όπου η πρόσβαση στο διαδίκτυο ή η ασύρματη επικοινωνία είναι ευκολότερη και πιο διαδεδομένη γίνεται πιο εύκολη η ανάπτυξη συστημάτων και εφαρμογών που αφορούν την παρακολούθηση ενός χώρου μέσω αισθητήρων. Πολλές φορές αυτά τα συστήματα τοποθετούνται σε ειδικούς χώρους ή ακόμα και σε σπίτια για να εξυπηρετήσουν κάποια λειτουργία ή να επιβλέπουν τον χώρο. Σε συνδυασμό με την τεχνολογική άνοδο των κινητών, των εφαρμογών τους και της επικοινωνίας μεταξύ αυτών και των συσκευών δίνεται η δυνατότητα στην πιο γρήγορη και εύκολη υλοποίηση έργων που αφορούν την επικοινωνία αισθητήρων με ένα κινητό και την επίβλεψη αυτών μέσω του τοπικού δικτύου ή διαδικτύου..

Η εργασία αυτή αφορά τη μελέτη και ανάπτυξη android εφαρμογής παρακολούθησης περιβαλλοντικών συνθηκών με raspberry pi 3. Η εφαρμογή android επικοινωνεί με το raspberry μέσω wifi για την παρακολούθηση θερμοκρασίας και υγρασίας σε έναν χώρο. Ο χρήστης μπορεί να ενεργοποιεί την φορητή συσκευή μικρού κόστους και κατανάλωσης και να επιβλέπει με το κινητό του τις περιβαντολλογικές συνθήκες πολύ γρήγορα. Η χρησιμότητα της εφαρμογής είναι η καταμέτρηση της θερμοκρασίας/υγρασίας σε διάφορα σημεία και ύψη ενός σπιτιού ή ενός χώρου. Επίσης, παρέχεται η δυνατότητα προσθήκης περισσότερων κόμβων που μπορεί να διαχειριστεί η εφαρμογή. Τέλος, μια πολύ σημαντική δυνατότητα είναι η επίβλεψη των τιμών των αισθητήρων διαδικτυακά μέσω της υπηρεσίας Thingspeak. Κάθε κόμβος μπορεί να στέλνει τα δεδομένα του μέσω ασφαλούς καναλιού στο Thingspeak και ο χρήστης μέσω της εφαρμογής να τα επιβλέπει από το κινητό του.

Ο κύριος στόχος αυτής της εργασίας είναι να μπορεί ο χρήστης με την εφαρμογή στο κινητό να μπορεί να επιβλέπει κόμβους με αισθητήρες που έχουν τοποθετηθεί σε έναν χώρο. Οι κόμβοι και το κινητό πρέπει να είναι συνδεδεμένοι στο ίδιο τοπικό δίκτυο.

Επιπλέον παρέχεται η δυνατότητα επίβλεψης των κόμβων μέσω διαδικτύου μέσω της υπηρεσίας Thingspeak και η προσθήκη, επεξεργασία και διαγραφή νέων κόμβων.

1.2 Δομή της εργασίας

Στο πρώτο κεφάλαιο παρουσιάζεται μια μικρή εισαγωγή της εργασίας και οι στόχοι της.

Στο δεύτερο κεφάλαιο παρουσιάζεται η εισαγωγή στο σύστημα παρακολούθησης και καταγραφής.

Στο τρίτο κεφάλαιο περιγράφεται η τεχνολογία και τα εργαλεία που χρησιμοποιήθηκαν για να υλοποιηθεί το έργο.

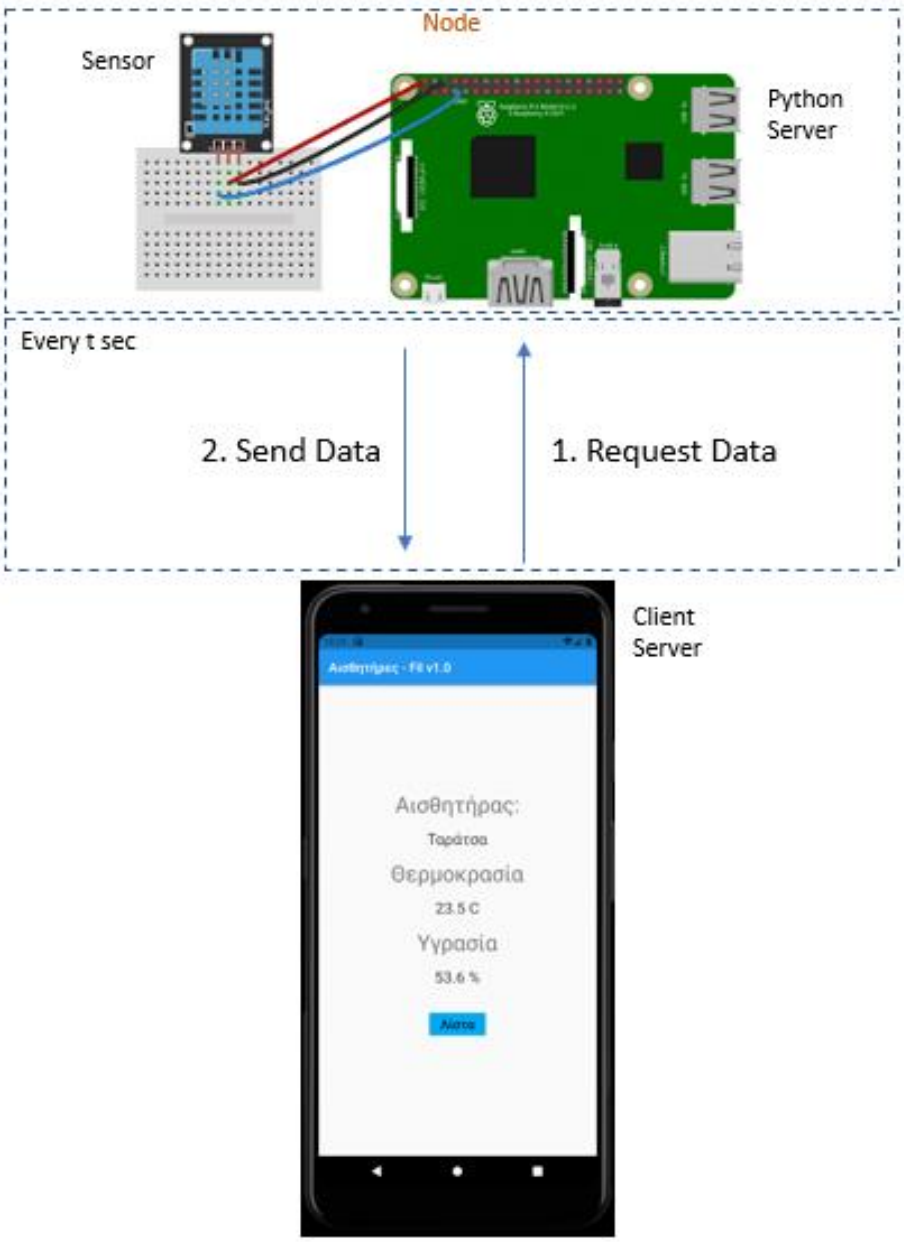
Στο τέταρτο κεφάλαιο αναλύονται τα προγράμματα στο Raspberry και η εφαρμογή στο κινητό με διαγράμματα και επεξηγήσεις.

Στο πέμπτο κεφάλαιο παρουσιάζονται τα συμπεράσματα της εργασίας και θέματα για μελλοντική έρευνα ενώ στο τέλος της εργασίας παρατίθεται το παράρτημα με κάποιους από τους κώδικες που χρησιμοποιήθηκαν στην εργασία.

Κεφάλαιο 2ο: Εισαγωγή στο σύστημα παρακολούθησης

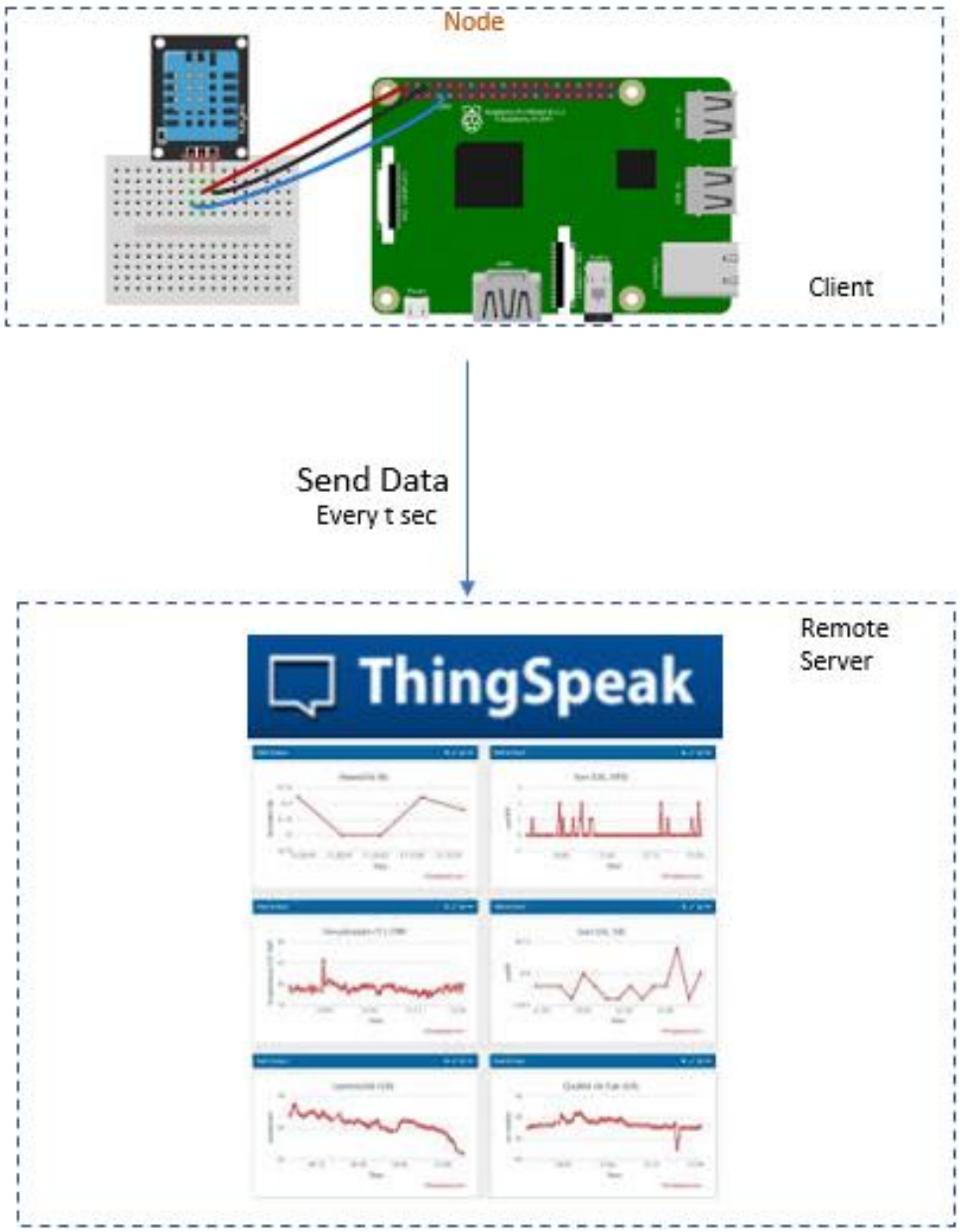
2.1 Εισαγωγή

Το σύστημα που υλοποιήθηκε ανάμεσα στην εφαρμογή στο κινητό και στο raspberry παρουσιάζεται στην εικόνα 2.1.



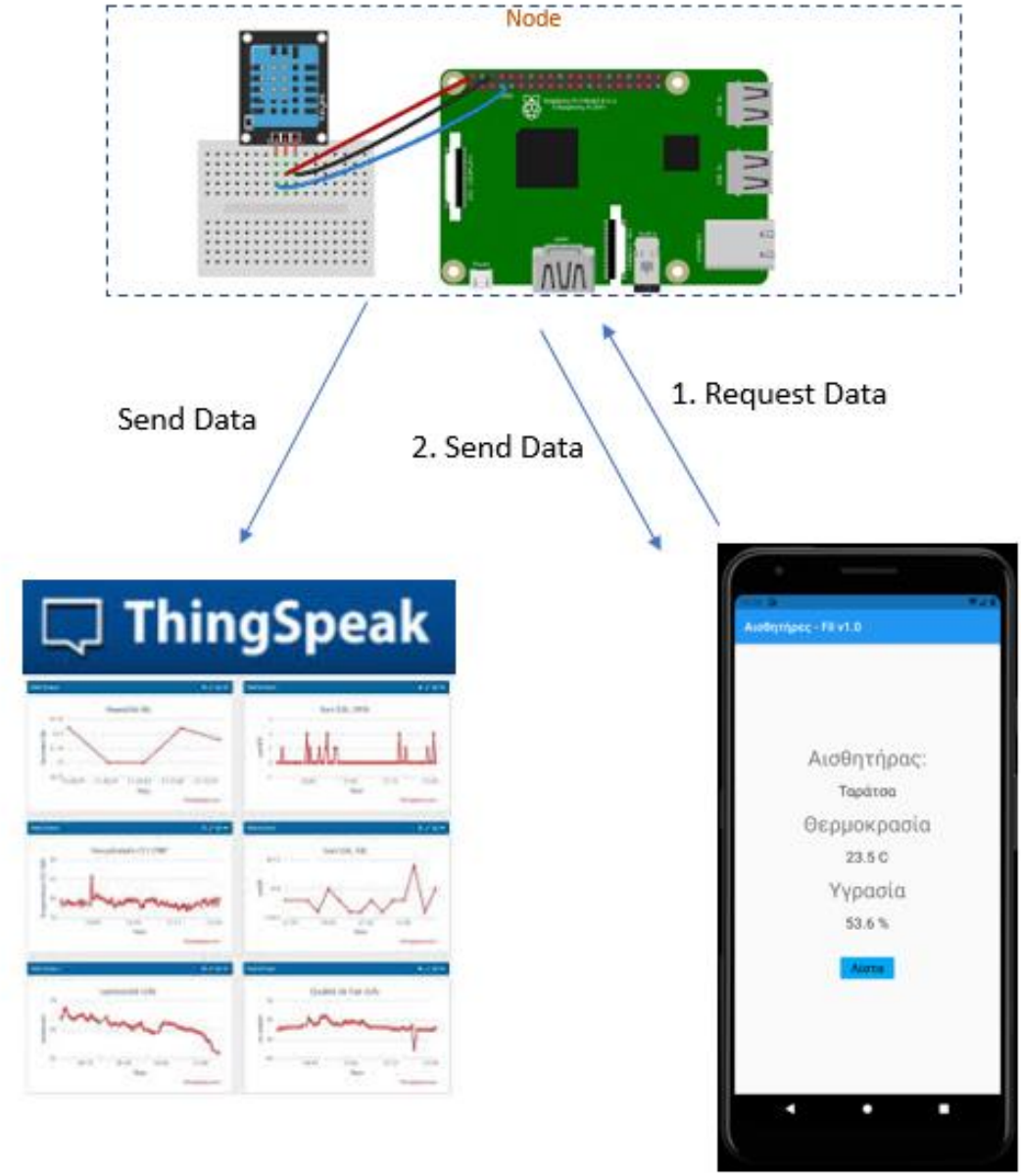
Εικόνα 2.1: Επικοινωνία εφαρμογής στο κινητό με τον κόμβο

Η εφαρμογή στο κινητό ζητάει κάθε t δευτερόλεπτα από τον κόμβο-raspberry να του επιστρέψει τις τιμές που διάβασε από τους αισθητήρες. Για να πραγματοποιηθεί αυτό θα πρέπει η εφαρμογή στο κινητό να γνωρίζει τη διεύθυνση του κόμβου.



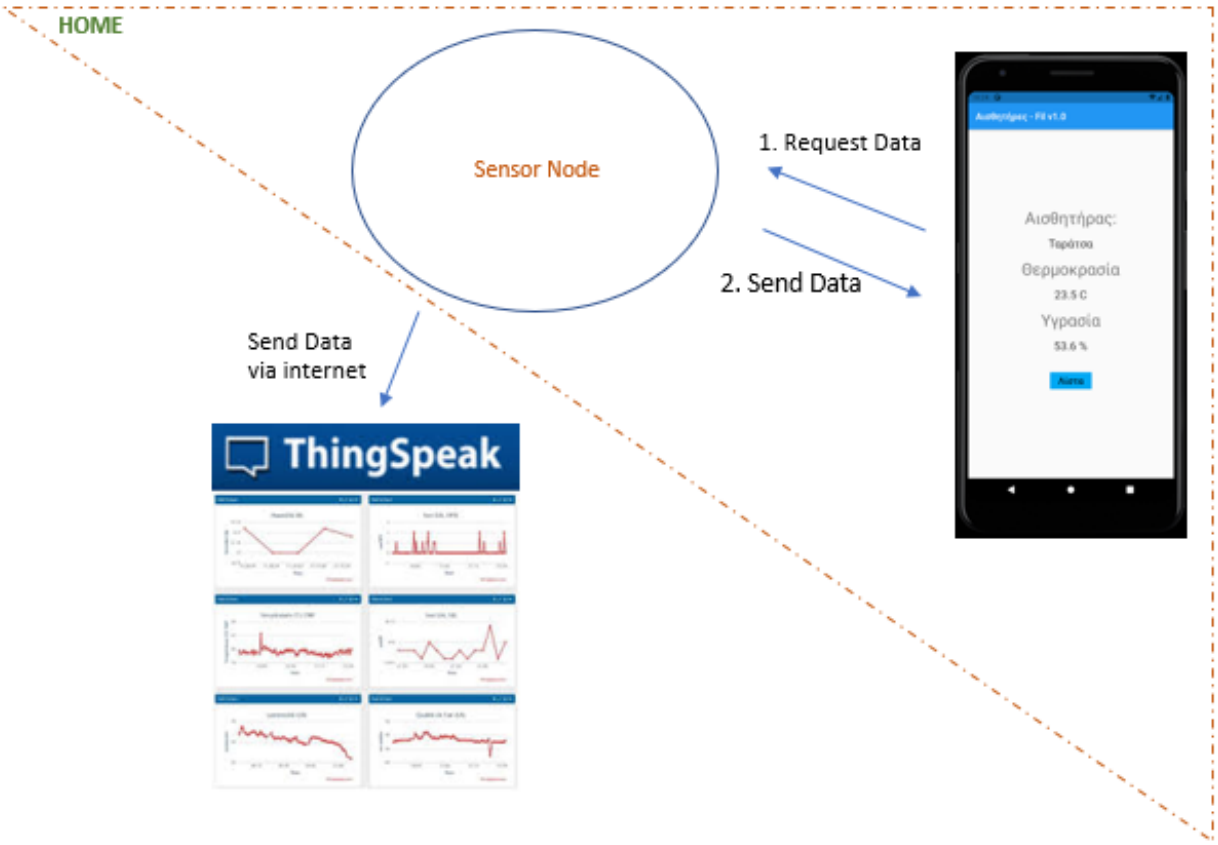
Εικόνα 2.2: Επικοινωνία κόμβου με το Thingspeak

Όπως φαίνεται στην Εικόνα 2.2. κάθε t δευτερόλεπτα ο κόμβος-raspberry στέλνει τις τιμές που διάβασε από τους αισθητήρες στο Thingspeak, ο οποίος είναι ένας remote server και λεπτομέρειες για αυτόν θα παρουσιαστούν σε επόμενο κεφάλαιο. Για να πραγματοποιηθεί αυτό θα πρέπει η εφαρμογή στο raspberry να γνωρίζει τη διεύθυνση του καναλιού στο Thingspeak.



Εικόνα 2.3: Επικοινωνία κόμβου με το Thingspeak και με την εφαρμογή στο κινητό

Στην Εικόνα 2.3. παροσυσίαζεται το σύστημα με τις δύο κύριες λειτουργίες επικοινωνίας , όπου κάθε t δευτερόλεπτα ο κόμβος-raspberry στέλνει τις τιμές που διάβασε από τους αισθητήρες στο Thingspeak και η εφαρμογή στο κινητό ζητάει κάθε t δευτερόλεπτα από τον κόμβο-raspberry να του επιστρέψει τις τιμές που διάβασε από τους αισθητήρες.

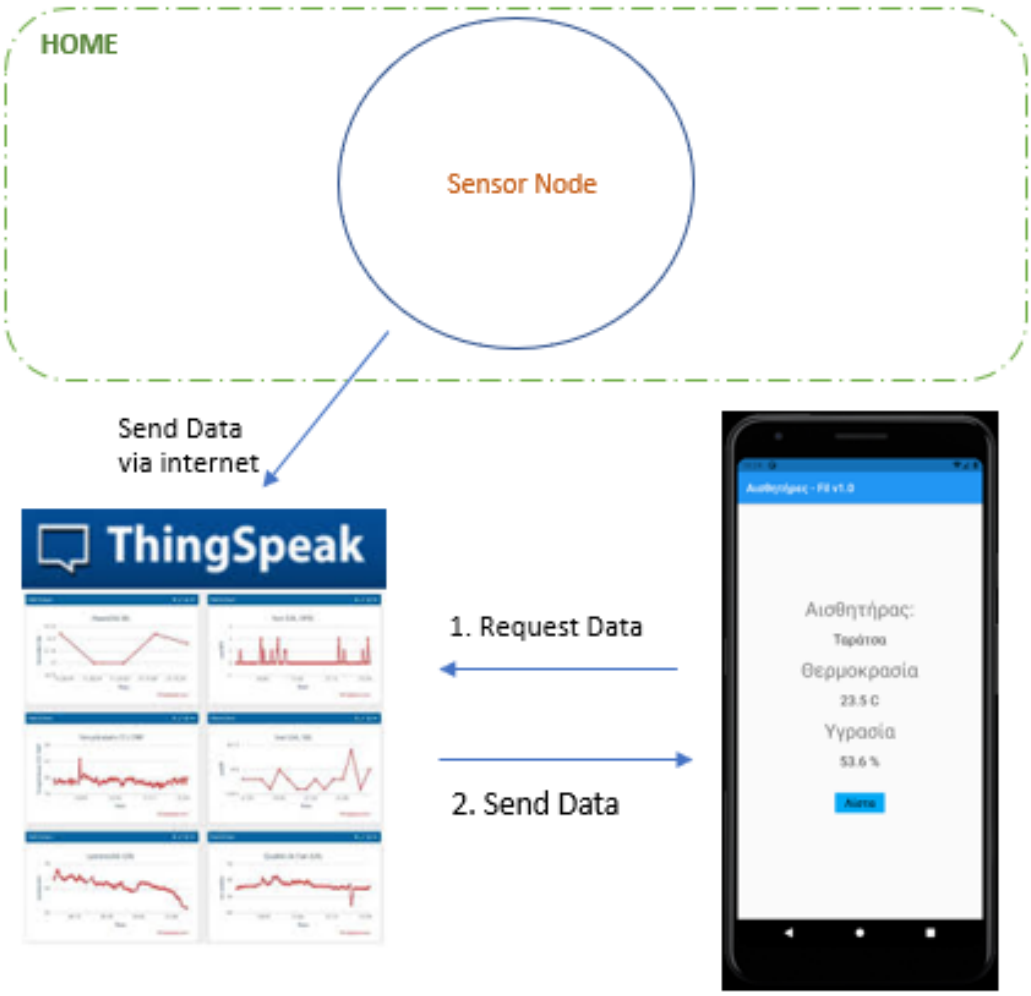


Εικόνα 2.4: Η επικοινωνία σε ένα σπίτι όταν το κινητό βρίσκεται εντός τοπικού δικτύου

Στην Εικόνα 2.4 παρουσιάζεται ο τρόπος επικοινωνίας όταν ο κόμβος τοποθετείται σε ένα σπίτι ή ένα χώρο όπου το τοπικό δίκτυο βρίσκεται πίσω από κάποιο Router ή firewall. Η εφαρμογή μπορεί να επικοινωνήσει με τον κόμβο όσο βρίσκονται και οι δύο στο ίδιο τοπικό δίκτυο. Σε περίπτωση που το κινητός βρίσκεται εκτός ίδιου τοπικού δικτύου, όπως στην εικόνα 2.5 τότε η εφαρμογή στο κινητό δεν μπορεί να επικοινωνήσει με τον κόμβο το κινητό βρίσκεται έξω από το σπίτι και ο κόμβος μέσα σε αυτόν. Το κινητό μπορεί να επικοινωνήσει με τον κόμβο μόνο αν γνωρίζει την τοπική διεύθυνση,

Ένας τρόπος για να επιτευχθεί επιτυχώς η επικοινωνία είναι να υλοποιηθεί port-forwarding στον δρομολογητή, που σημαίνει ότι πρέπει να πραγματοποιηθεί ‘επέμβαση’ στο router, ώστε να αντιστοιχηθεί η εξωτερική ip του σπιτιού με αυτή του κόμβου σε κάποιο συγκεκριμένο port.

Η επικοινωνία του κόμβου με το Thingspeak μπορεί να γίνει χωρίς port-forwarding αφού ο κόμβος στέλνει τιμές.

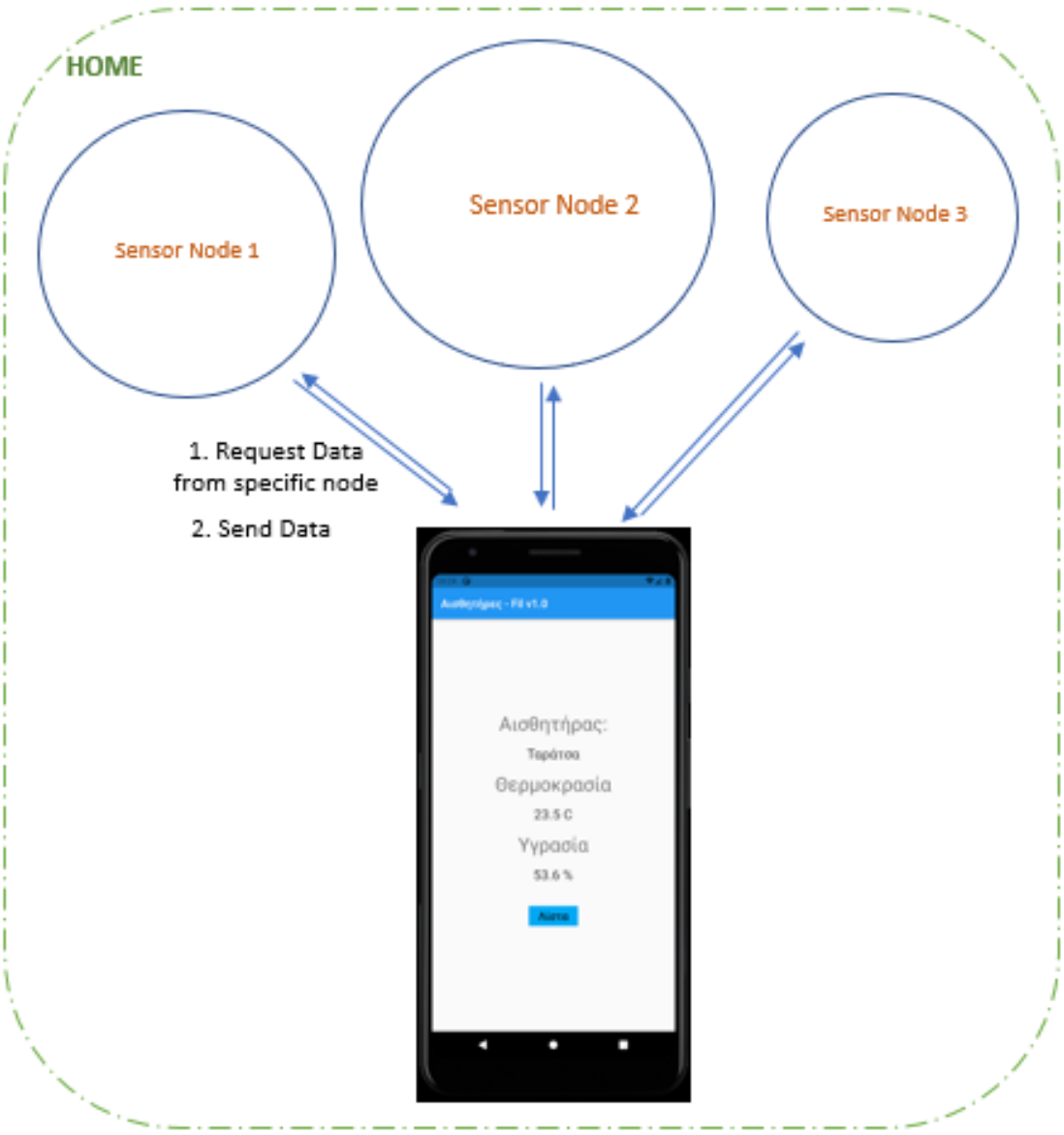


Εικόνα 2.5: Η επικοινωνία σε ένα σπίτι όταν το κινητό βρίσκεται εκτός τοπικού δικτύου

Ένας άλλος τρόπος για να επιτευχθεί επιτυχώς η επικοινωνία της εφαρμογής με τον κόμβο όταν δεν βρίσκονται στον ίδιο δίκτυο είναι η επικοινωνία της εφαρμογής με το Thingspeak, χωρίς να επικοινωνεί άμεσα με τον κόμβο.

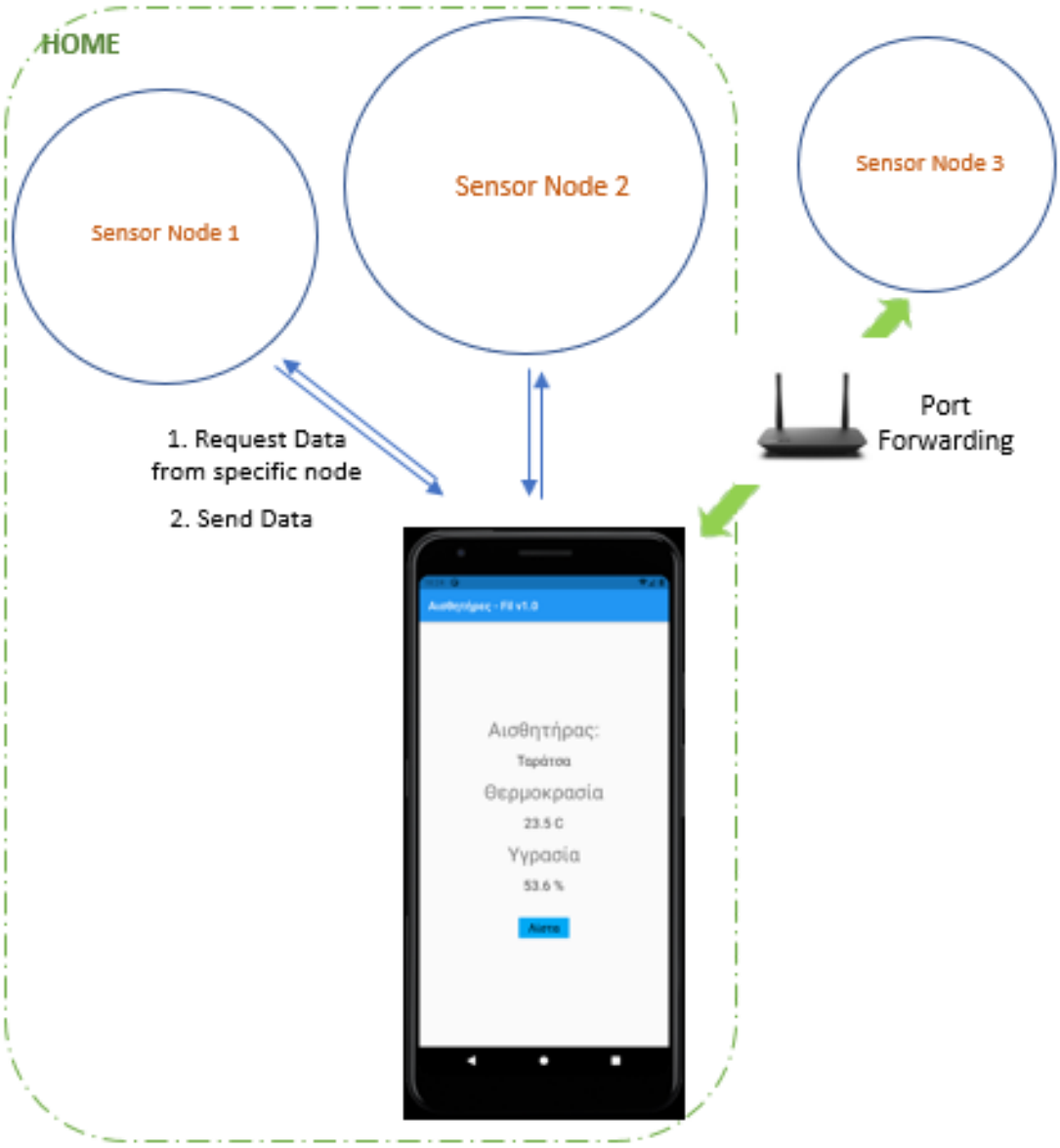
Το Thingspeak λαμβάνει τις τιμές συνεχώς και η εφαρμογή μπορεί να τις λαμβάνει από τον server.

Έχουν υλοποιηθεί και οι δύο τρόποι στην εφαρμογή.



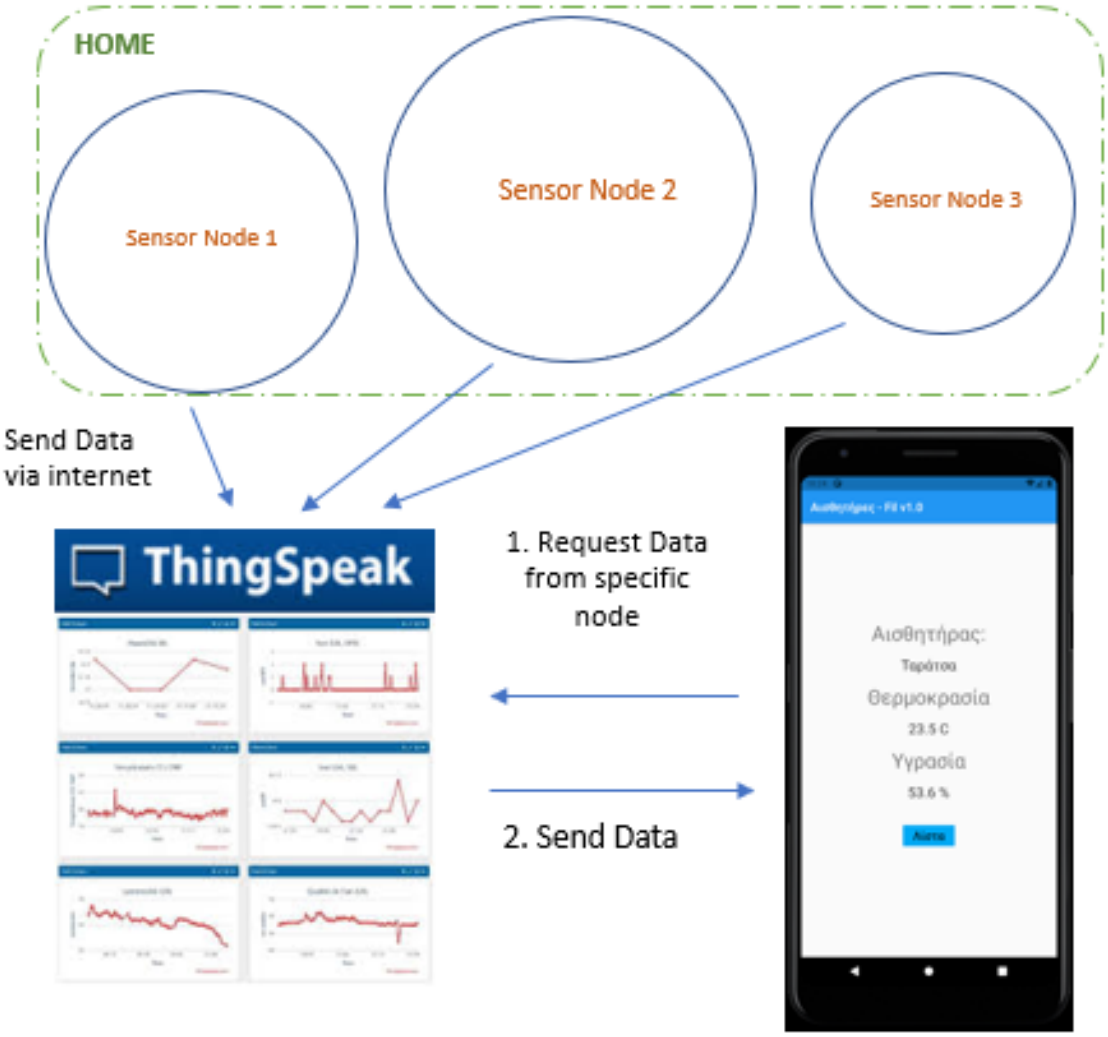
Εικόνα 2.6: Το σύστημα – επικοινωνία εφαρμογής με πολλούς κόμβους σε εσωτερικό δίκτυο

Στην Εικόνα 2.6 παρουσιάζεται η δυνατότητα της εφαρμογής να επικοινωνεί με όσους κόμβους υπάρχουν σε ένα εσωτερικό χώρο, πχ σπίτι (στο ίδιο δίκτυο).



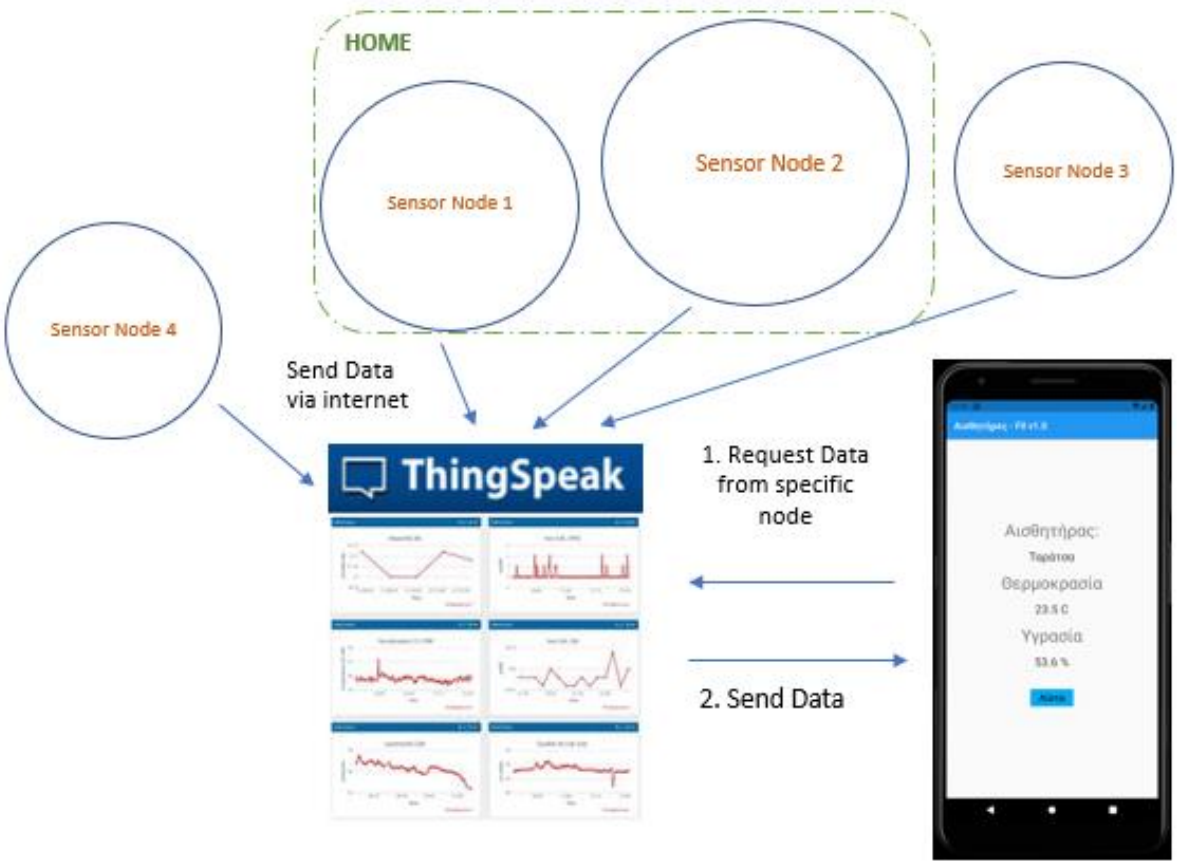
Εικόνα 2.7: Επικοινωνία εφαρμογής με πολλούς κόμβους σε εσωτερικό δίκτυο και με ένας εκτός

Στην Εικόνα 2.7 φαίνεται ότι όποιος δεν κόμβος βρίσκεται μέσα στο ίδιο τοπικό δίκτυο θα πρέπει να πραγματοποιηθεί port-forwarding.



Εικόνα 2.8: Τεχνική επικοινωνία εφαρμογής με πολλούς κόμβους όταν βρίσκεται εκτός εσωτερικού δικτύου

Στην Εικόνα 2.8 παρουσιάζεται η δυνατότητα της εφαρμογής να επικοινωνεί με όσους κόμβους υπάρχουν στον εσωτερικό χώρο (πίσω από ένα router-firewall) από το διαδίκτυο μέσω του Thingspeak.



Εικόνα 2.9: Με τη βοήθεια του Thingspeak η εφαρμογή μπορεί να επιβλέπει κόμβους εκτός και εντός δικτύου

Στην Εικόνα 2.9 παρουσιάζεται η δυνατότητα της εφαρμογής να επικοινωνεί με όσους κόμβους υπάρχουν στον εσωτερικό αλλά και σε εξωτερικό χώρο μέσω του Thingspeak. Μπορούν όσοι κόμβοι έχουν σύνδεση στο διαδίκτυο και τη διεύθυνση του server να στέλνουν τα δεδομένα τους σε αυτόν.

Κεφάλαιο 3ο: Τεχνολογία και εργαλεία

3.1 Raspberry

Το Raspberry Pi 3 είναι ένας πλήρης υπολογιστής σε μέγεθος πιστωτικής κάρτας. Πέρα από τον μικρό όγκο του το Raspberry Pi 3 διαθέτει έναν τετραπύρηνο επεξεργαστή 1200MHz, μια διπύρηνη κάρτα γραφικών με GPU, 1GB RAM, τέσσερις θύρες USB, έχει έξοδο HDMI, τροφοδοτείται μέσω Micro USB και έχει 40 pins γενικής χρήσης για σύνδεση με άλλες ηλεκτρονικές συσκευές και περιφερειακά. Επίσης, υποστηρίζει μια οθόνη και με ένα πληκτρολόγιο και ποντίκι γίνεται ένας υπολογιστής με λειτουργικό σύστημα (όπως το Linux).

Τα βασικά χαρακτηριστικά του Raspberry Pi 3 είναι:

SoC: Broadcom BCM2837

CPU: 4× ARM Cortex-A53, 1.2GHz

GPU: Broadcom VideoCore IV

RAM: 1GB LPDDR2 (900 MHz)

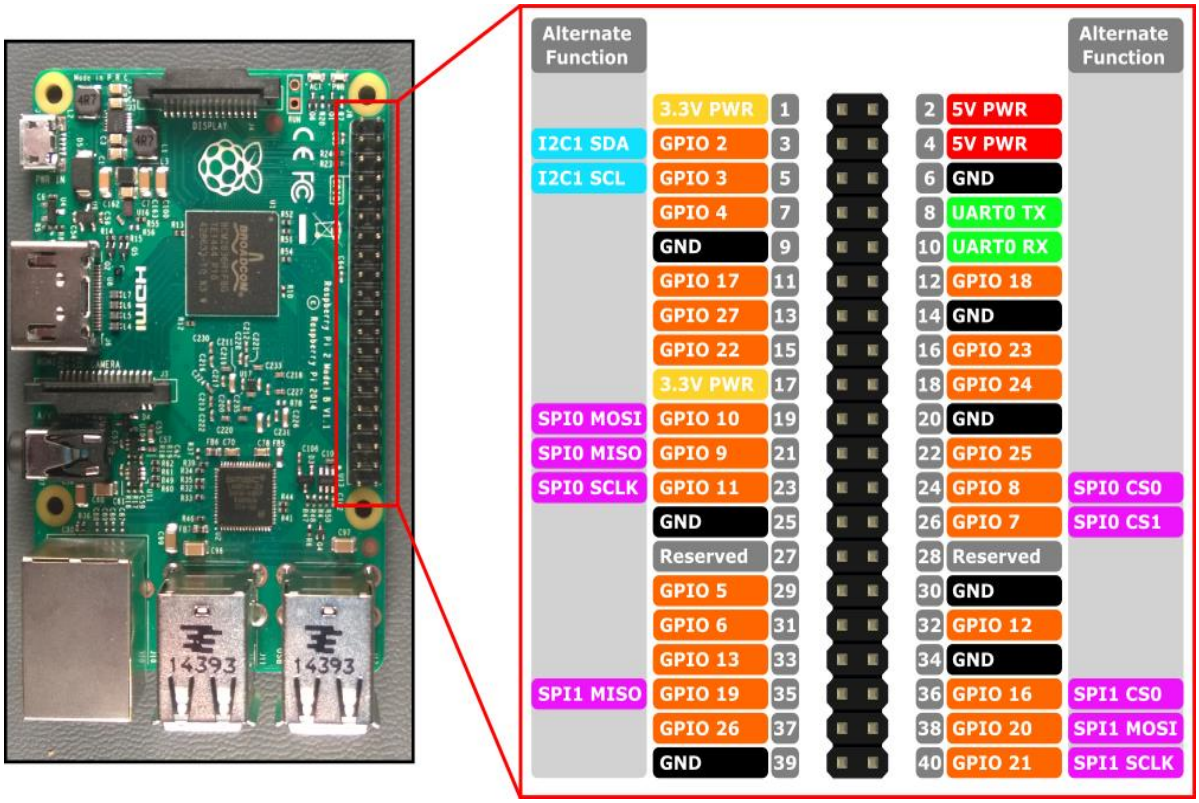
Networking: 10/100 Ethernet, 2.4GHz 802.11n wireless

Bluetooth: Bluetooth 4.1 Classic, Bluetooth Low Energy

Storage: microSD

GPIO: 40-pin header, populated

Ports: HDMI, 3.5mm analogue audio-video jack, 4× USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)



Εικόνα 3.1: Raspberry Pi 3 με τους ακροδέκτες

Για την υλοποίηση της εργασίας χρησιμοποιήθηκε το Raspberry Pi 3 (κόστος 39 ευρώ) και το μικρότερο μοντέλο Raspberry Pi Zero W (κόστος 10 ευρώ).

Εικόνα 3.2: Raspberry Pi Zero W

Τα χαρακτηριστικά του Raspberry Pi Zero W είναι:

- 802.11 b/g/n wireless LAN
- Bluetooth 4.1
- Bluetooth Low Energy (BLE)
- 1GHz, single-core CPU
- 512MB RAM
- Mini HDMI and USB On-The-Go ports
- Micro USB power
- HAT-compatible 40-pin header
- Composite video and reset headers
- CSI camera connector

Έχουν κυκλοφορήσει αρκετές γενιές Raspberry Pis. Τα Raspberry Pi διαθέτουν ένα σύστημα Broadcom σε ένα τσιπ (SoC) με ενσωματωμένη κεντρική μονάδα επεξεργασίας συμβατή με ARM (CPU) και μονάδα επεξεργασίας γραφικών on-chip (GPU), ενώ το Raspberry Pi Pico διαθέτει σύστημα RP2040 σε τσιπ με ενσωματωμένο ARM -συμβατή κεντρική μονάδα επεξεργασίας (CPU).

Η πρώτη γενιά (Raspberry Pi Model B) κυκλοφόρησε τον Φεβρουάριο του 2012, ακολουθούμενη από το απλούστερο και φθηνότερο Model A. Το 2014, κυκλοφόρησε μια πλακέτα με βελτιωμένο σχεδιασμό, το Raspberry Pi Model B +. Αυτοί διαθέτουν επεξεργαστές ARM11, έχουν μέγεθος περίπου πιστωτικής κάρτας και αντιπροσωπεύουν τον βασικό κορμό. Τα βελτιωμένα μοντέλα A + και B + κυκλοφόρησαν ένα χρόνο αργότερα.

Το Raspberry Pi 2 κυκλοφόρησε τον Φεβρουάριο του 2015 και αρχικά παρουσίασε επεξεργαστή τετραπύρηνου ARM Cortex-A7 900 MHz 32-bit με RAM 1 GB. Οι νεότερες εκδόσεις περιλάμβαναν επεξεργαστή τετραπύρηνου ARM Cortex-A53 1,2 GHz 1,2 bit.

Το Raspberry Pi Zero με μικρότερο μέγεθος και μειωμένες δυνατότητες εισόδου/εξόδου (I/O) και γενικής χρήσης input/output (GPIO) κυκλοφόρησε τον Νοέμβριο του 2015. Στις 28 Φεβρουαρίου 2017, κυκλοφόρησε το Raspberry Pi Zero W, μια έκδοση του Zero με δυνατότητες Wi-Fi και Bluetooth, για 10 δολλάρια. Στις 12 Ιανουαρίου 2018, κυκλοφόρησε το Raspberry Pi Zero WH, μια έκδοση του Zero W με προ-κολλημένους ακροδέκτες GPIO.

Το Raspberry Pi 3 Model B κυκλοφόρησε τον Φεβρουάριο του 2016 με επεξεργαστή τετραπύρηνου ARM Cortex-A53 1,2 GHz, 1.2 GHz, ενσωματωμένο Wi-Fi 802.11n, δυνατότητα εκκίνησης Bluetooth και USB. Λίγο αργότερα κυκλοφόρησε το Raspberry Pi 3 Model B + με έναν γρηγορότερο επεξεργαστή 1,4 GHz, ένα τριπλάσιο ταχύτερο Gigabit Ethernet και 2,4 / 5 GHz dual-band 802.11ac Wi-Fi (100 Mbit /s). Άλλες πολύ χρήσιμες λειτουργίες είναι το Power over Ethernet (PoE), η εκκίνηση USB και η εκκίνηση δικτύου (που σημαίνει ότι δεν απαιτείται πλέον κάρτα SD).

Το Raspberry Pi 4 Model B κυκλοφόρησε τον Ιούνιο του 2019 με επεξεργαστή τετραπύρηνου ARM Cortex-A72 1,5 GHz, ενσωματωμένο Wi-Fi 802.11ac, Bluetooth 5, πλήρης gigabit Ethernet, δύο Θύρες USB 2.0, δύο θύρες USB 3.0 και υποστήριξη διπλής οθόνης μέσω ζεύγους θυρών micro HDMI για ανάλυση έως 4K. Το Pi 4 τροφοδοτείται επίσης μέσω θύρας USB-C, επιτρέποντας την παροχή πρόσθετης ισχύος στα περιφερειακά που την χρειάζονται. Η αρχική πλακέτα Raspberry Pi 4 έχει σχεδιαστικό ελάττωμα, όπου καλώδια USB τρίτων κατασκευαστών, όπως αυτά που χρησιμοποιούνται

σε Apple MacBooks, το αναγνωρίζουν εσφαλμένα και αρνούνται να παράσχουν ισχύ. Το ελάττωμα του σχεδιασμού διορθώθηκε στην αναθεώρηση της έκδοσης 1.2 που κυκλοφόρησε στα τέλη του 2019.

Το Raspberry Pi 400 κυκλοφόρησε τον Νοέμβριο του 2020. Διαθέτει προσαρμοσμένη πλακέτα που προέρχεται από το υπάρχον Raspberry Pi 4, ειδικά αναδιαμορφωμένο με δικό του πληκτρολόγιο. Διαθέτει ψύξη ώστε να επιτρέπει στον επεξεργαστή Broadcom BCM2711C0 του Raspberry Pi 400 να ανέβει στα 1,8 GHz, το οποίο είναι ελαφρώς υψηλότερο από το Raspberry Pi 4 στο οποίο βασίζεται.

Το Raspberry Pi Pico κυκλοφόρησε τον Ιανουάριο του 2021 με τιμή λιανικής 4 δολλάρια. Ήταν η πρώτη πλακέτα του Raspberry Pi βασισμένη σε ένα μόνο τσιπ μικροελεγκτή. το RP2040, το οποίο σχεδιάστηκε από το Raspberry Pi. Το Pico διαθέτει 264 KB μνήμης RAM και 2 MB μνήμης flash. Είναι προγραμματιζόμενο σε MicroPython, CircuitPython και C. Έχει σχεδιαστεί σε συνεργασία με Adafruit, Pimoroni, Arduino και Sparkfun για την κατασκευή των εξαρτημάτων του Το πιο σημαντικό χαρακτηριστικό του είναι ότι είναι σχεδιασμένος για φυσική χρήση υπολογιστών, παρόμοιος με την έννοια του Arduino.

Όπως αναφέρθηκε το Raspberry Pi 3 συνδέοντάς το σε μια οθόνη και προσθέτοντας πληκτρολόγιο και ποντίκι είναι ένας πλήρης υπολογιστής, ο οποίος υποστηρίζει κάποιες από τις διανομές Linux.

Μερικές από τις διανομές που υποστηρίζονται είναι:

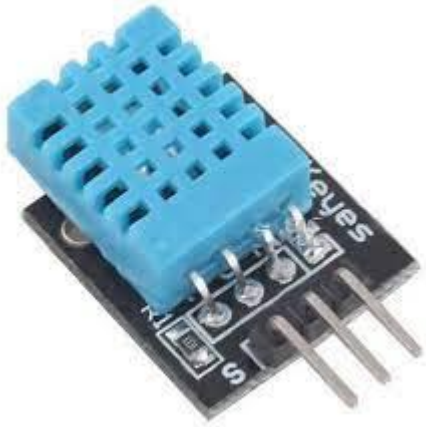
- Raspbian
- Windows 10 IoT Core
- Ubuntu Core
- Kali Linux
- FreeBSD

Το Raspberry Pi έχει δύο συγκεκριμένες λειτουργίες μέσω των ακροδεκτών του, να μπορεί να γίνει έξοδος ή είσοδος. Χρησιμοποιώντας αυτούς τους ακροδέκτες μπορεί να χρησιμοποιηθεί ένας ψηφιακός αισθητήρας αλλά όταν χρειάζεται να χρησιμοποιηθεί αναλογικός αισθητήρας θα πρέπει να χρησιμοποιηθούν ειδικοί μετατροπείς αναλογικού σε ψηφιακό σήμα.

3.2 Αισθητήρες

Το σύστημα μέτρησης θερμοκρασίας-υγρασίας έχει σαν βασικό περιφερειακό ένα ψηφιακό αισθητήριο χαμηλής ακρίβειας για να μην χρειαστεί να χρησιμοποιηθούν οι μετατροπείς αναλογικού σε ψηφιακό. Δεν χρειάζεται να προστεθούν παραπάνω αισθητήρια γιατί ο σκοπός της εργασίας είναι η επικοινωνίας της εφαρμογής στο κινητό και του Thingspeak με το Raspberry που διαθέτει το αισθητήριο.

Χρησιμοποιήθηκε ο αισθητήρας DHT11 που μπορεί να μετρήσει την υγρασία και τη θερμοκρασία.



Εικόνα 3.3: Αισθητήρας DTH11

3.3 Flutter

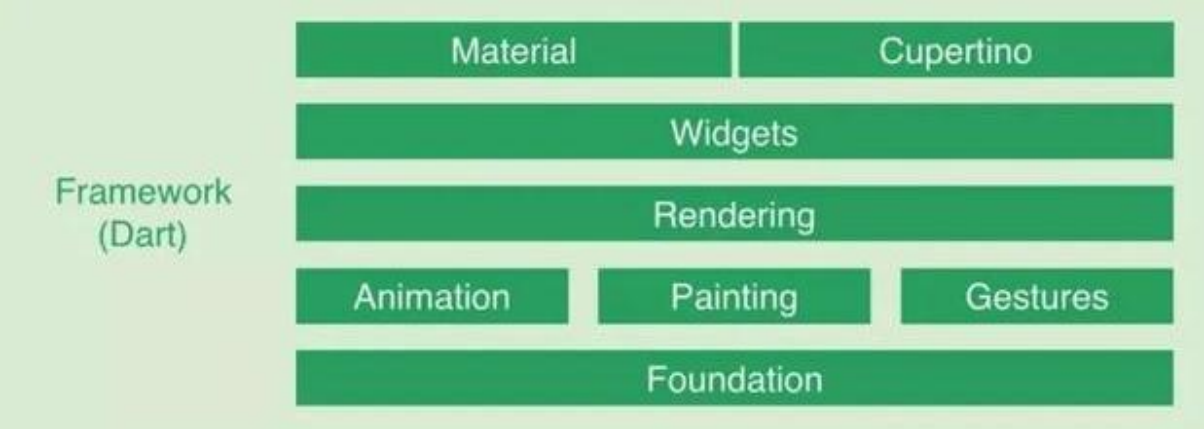
Το Flutter παρουσιάστηκε από την Google ως τεχνολογία ανοιχτού κώδικα για την κωδικοποίηση και τη δημιουργία εγγενών(native) εφαρμογών για Android και iOS. Το Flutter είναι σχετικά νέο καθώς παρουσιάστηκε επίσημα τον Δεκέμβριο του 2018 ως η πρώτη σταθερή έκδοση 1.0 στο Flutter Live event.

Το Flutter συνδυάζει ευκολία ανάπτυξης με απόδοση παρόμοια με την εγγενή απόδοση, διατηρώντας παράλληλα την οπτική συνέπεια μεταξύ των πλατφορμών. Η γλώσσα προγραμματισμού του Flutter, Dart, προοριζόταν αρχικά ως αντικατάσταση του JavaScript. Το πιο σημαντικό, το Flutter είναι ανοιχτού κώδικα και εντελώς δωρεάν. Προς το παρόν, το Flutter έχει ίση δημοτικότητα με το React Native τόσο στο GitHub όσο και στο Stack Overflow.

Υπάρχουν ήδη πάνω από 40.000 εφαρμογές Flutter στο Google Play Store και αυτός ο αριθμός αυξάνεται με υψηλό ρυθμό. Οι eBay και Alibaba Group και άλλοι δημοφιλείς πάροχοι ηλεκτρονικού εμπορίου χρησιμοποιούν επίσης το Flutter για να δώσουν ομοιόμορφη εμφάνιση στις εφαρμογές ιστού και κινητών.

Το Flutter τείνει να γίνει μια ισχυρή, γενικής χρήσης, ανοιχτή εργαλειοθήκη διεπαφής χρήστη για τη δημιουργία εμπειριών σε οποιαδήποτε ενσωματωμένη συσκευή, κινητό και υπολογιστή.

Το πλαίσιο του Flutter είναι γραμμένο στη γλώσσα προγραμματισμού Dart, διαθέτει τη μηχανή Flutter, τη βιβλιοθήκη Foundation και widget. Η προσέγγιση της ανάπτυξης στο Flutter διαφέρει από τις υπόλοιπες παρόμοιες τεχνολογίες από τη δηλωτική γραφή της διεπαφής χρήστη. Στο Flutter ο προγραμματιστής-σχεδιαστής μπορεί να ξεκινήσει από το τέλος, δηλαδή να ξεκινήσει την ανάπτυξη κάποιου στοιχείου. Πολλοί προγραμματιστές διακρίνουν αυτήν την ανάπτυξη διεπαφής χρήστη ως πιο καλή, αλλά προκαλεί επίσης ορισμένες δυσκολίες για τους προγραμματιστές στην αρχή.



Εικόνα 3.4: Flutter Framework

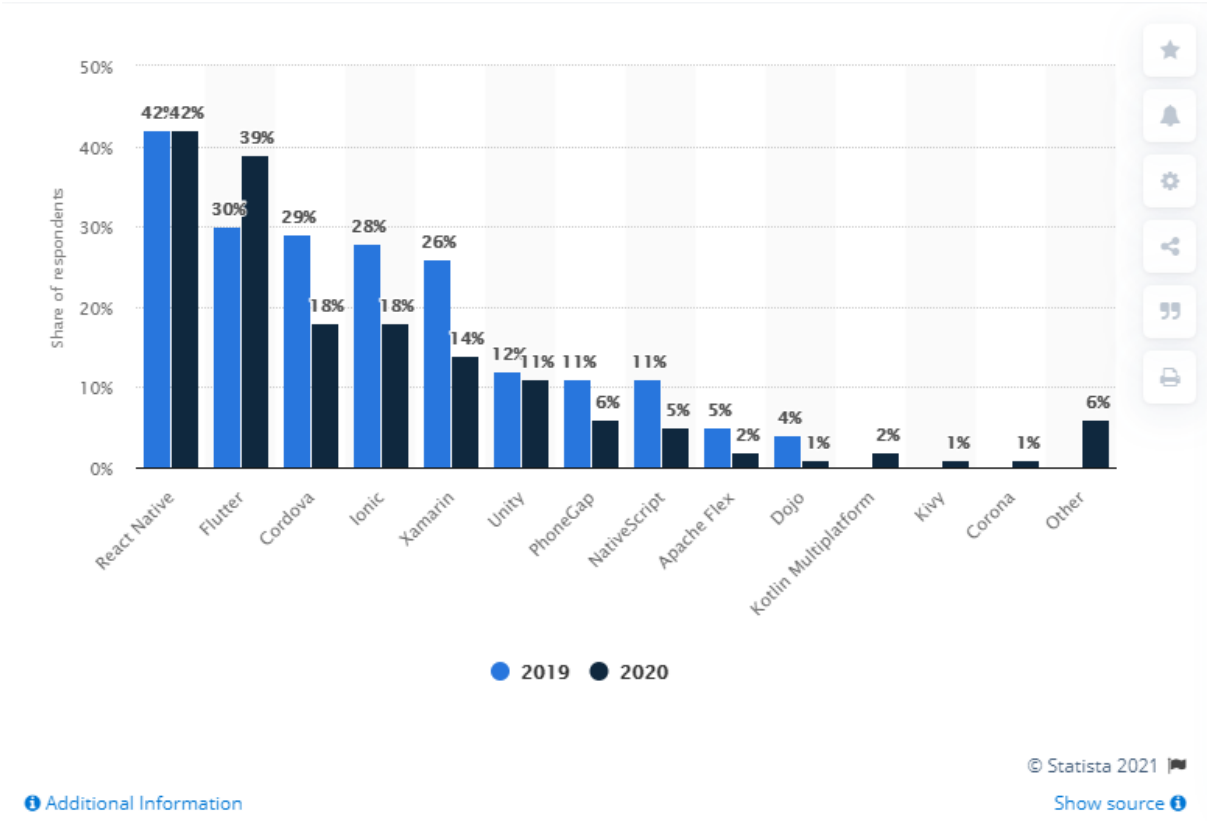
Η κύρια ιδέα του Flutter είναι ότι οι προγραμματιστές μπορούν να δημιουργήσουν ολόκληρο το περιβάλλον εργασίας χρήστη συνδυάζοντας απλά διαφορετικά widgets. Η διεπαφή της εφαρμογής αποτελείται από διάφορα ένθετα widget, τα οποία μπορεί να είναι οποιοδήποτε αντικείμενο. Αυτό ισχύει για οτιδήποτε, από τα κουμπιά έως την επένδυση και συνδυάζοντας widget, ο προγραμματιστής μπορεί να προσαρμόσει ριζικά την εφαρμογή. Τα widget μπορούν να επηρεάσουν το ένα το άλλο και να χρησιμοποιούν ενσωματωμένες λειτουργίες για να ανταποκρίνονται σε εξωτερικές αλλαγές στην κατάσταση. Τα widget είναι σημαντικά στοιχεία της διεπαφής χρήστη και συμμορφώνονται με τις προδιαγραφές σχεδιασμού Android, iOS και συμβατικών εφαρμογών ιστού.

Με το Flutter, οι προγραμματιστές μπορούν να δημιουργήσουν προσαρμοσμένα widget, τα οποία μπορούν εύκολα να συνδυαστούν με τα υπάρχοντα. Δεν υπάρχουν widget έτοιμα από κάποιον κατασκευαστή (τρίτο) , αλλά το Flutter παρέχει στους προγραμματιστές τα δικά τους έτοιμα widget - ένα σύνολο παραδειγμάτων εφαρμογών που δείχνουν πώς να χρησιμοποιούν τυπικά widget - που μοιάζουν με native γλώσσες σχεδίασης Android και iOS.

Το Flutter παρέχει επίσης στους προγραμματιστές τη δυνατότητα να βλέπουν widget σε αντιδραστικό στυλ. Δεν είναι το πρώτο που το κάνει, αλλά το Flutter είναι το μόνο SDK για κινητά που προσφέρει μια αντιδραστική εμφάνιση χωρίς την ανάγκη για γέφυρα JavaScript. Επιπλέον, το Dart έρχεται με ένα αποθετήριο πακέτων λογισμικού για να βελτιώσει τις δυνατότητες των εφαρμογών. Για παράδειγμα, προσφέρει πολλά πακέτα που βοηθούν στην πρόσβαση στο Firebase, έτσι ώστε οι προγραμματιστές να μπορούν να δημιουργούν εφαρμογές χωρίς διακομιστές.

Οφέλη του Flutter

Όσον αφορά την ανάπτυξη εφαρμογών, οι προγραμματιστές δεν περιορίζονται σε ένα ενιαίο πλαίσιο κινητής τηλεφωνίας. Στην πραγματικότητα, τα στατιστικά στοιχεία δείχνουν ότι ενώ η πλειονότητα των προγραμματιστών χρησιμοποιούν το React Native (42%), η χρήση του Flutter το 2020 (39%) έχει αυξηθεί πολύ σε σύγκριση με τη χρήση του Flutter το 2019 (30%).



Εικόνα 3.5: Cross-platform mobile frameworks used by software developers worldwide in 2019 and 2020 (https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours)

Οι δημιουργοί Flutter ήθελαν να εφεύρουν μια τεχνολογία με την ταχύτερη ευκαιρία να παραδώσουν μια εφαρμογή για κινητά πολλαπλών πλατφορμών με εξαιρετική απόδοση. Οι ακόλουθες δυνατότητες το επιτρέπουν:

Επαναφόρτωση – Hot Reload.

Η επαναφόρτωση του Flutter βοηθά στην εξοικονόμηση χρόνου κατά την ανάπτυξη, επιτρέποντας στον προγραμματιστή να δει τις εφαρμοζόμενες αλλαγές σε πραγματικό χρόνο. Αυτή η ικανότητα βοηθά τους προγραμματιστές να είναι πολύ πιο αποτελεσματικοί και παραγωγικοί. Επιτρέπει στον προγραμματιστή να θέσει σε παύση την εκτέλεση κώδικα, να κάνει αλλαγές στον κώδικα και να συνεχίσει τον κώδικα από το ίδιο μέρος. Αυτό επιταχύνει σημαντικά την ανάπτυξη και επιτρέπει περισσότερους πειραματισμούς.

Ένα από τα πιο σημαντικά οφέλη του Flutter είναι ο τρόπος με τον οποίο χρησιμοποιεί έτοιμα widget. Αυτό διασφαλίζει ότι το Flutter προσφέρει ένα συνεπές μοντέλο ανάπτυξης και σχεδιασμού. Τα γραφικά στοιχεία βασίζονται στην Google, επομένως έχουν υψηλή ποιότητα κώδικα και αποδίδουν καλύτερα από άλλα πλαίσια ανοιχτού κώδικα. Καθώς τα περισσότερα από αυτά είναι εξαιρετικά προσαρμόσιμα, εξοικονομούν χρόνο προγραμματιστή, όπως κανένα άλλο πλαίσιο. Εκτός από τα

βασικά widget διάταξης, τα widget Flutter ακολουθούν τόσο την εμφάνιση Material όσο και την εμφάνιση Cupertino, κάτι που είναι ένα τεράστιο πλεονέκτημα.

Ελάχιστος κωδικός και πρόσβαση σε native λειτουργίες. Το Flutter επιτρέπει στους προγραμματιστές να χρησιμοποιούν το Dart, το οποίο μεταγλωττίζεται απευθείας στον κώδικα ARM των κινητών συσκευών και βοηθά όχι μόνο στην επιτάχυνση των εφαρμογών, αλλά τους επιτρέπει να ξεκινήσουν και ταχύτερα.

Ένα ισχυρό ατού του Flutter είναι το Skia, ο ανοιχτός κώδικας, υψηλής απόδοσης μηχανισμός γραφικών που χρησιμοποιούν τα Adobe, Chrome και Amazon. Το Flutter επιτρέπει στους χρήστες να αναπτύξουν εφαρμογές με προσαρμοσμένα σχέδια, τα οποία θα φαίνονται εξίσου καλά σε συσκευές iOS και Android. Οι εφαρμογές που αναπτύχθηκαν στο Flutter - σε αντίθεση με τους ανταγωνιστές της - δεν διακατέχουν κανέναν κίνδυνο ότι θα υπάρξουν αποτυχίες UI κατά την ενημέρωση του λογισμικού.

Ένα τεράστιο πλεονέκτημα του Flutter είναι η δυνατότητα προσαρμογής όλων όσων βλέπει ο χρήστης στην οθόνη, ανεξάρτητα από την πολυπλοκότητα του στοιχείου. Η απαιτούμενη προσπάθεια είναι ουσιαστικά χαμηλότερη από αυτή που απαιτείται στο λογισμικό ανάπτυξης native πλατφορμών.

Το flutter έχει την καλύτερη απόδοση στην κατηγορία (cross-platform) και η κατανάλωση πόρων λόγω της συλλογής native κωδικών και υψηλής απόδοσης μηχανής απόδοσης. Ο πρώτος παρέχει έναν εύκολο τρόπο δημιουργίας επικοινωνίας μεταξύ native κώδικα πλατφόρμας και Dart μέσω καναλιών πλατφόρμας. Έτσι, οι προγραμματιστές μπορούν να εφαρμόσουν σε μια εφαρμογή Flutter οτιδήποτε μπορεί να κάνει μια εγγενής εφαρμογή, μόνο με λίγο περισσότερη προσπάθεια από την native πλευρά. Λόγω της μηχανής Skia ένα UI ενσωματωμένο στο Flutter μπορεί να ξεκινήσει σχεδόν σε οποιαδήποτε πλατφόρμα, υποθέτοντας ότι αυτή η πλατφόρμα υποστηρίζει το Flutter. Με διαφορετικό τρόπο, οι προγραμματιστές δεν χρειάζεται πλέον να προσαρμόζουν το περιβάλλον εργασίας χρήστη για να το μεταφέρουν σε μια πλατφόρμα, η οποία απλοποιεί τη διαδικασία ανάπτυξης σε μεγάλο βαθμό.

Η καλύτερη παραγωγικότητα του προγραμματιστή επιτυγχάνεται λόγω του ότι το Flutter έχει σχεδιαστεί κυρίως για ταχύτερη σύνταξη κώδικα. Αποτελείται από έτοιμα προς χρήση γραφικά στοιχεία, η σύνταξή του απαιτεί λιγότερο γράψιμο κώδικα και οι ταχύτητες επαναφόρτωσης αυξάνουν την αναζήτηση και διόρθωση σφαλμάτων. Όλα αυτά οδηγούν σε λιγότερες ανθρωποώρες για προγραμματιστές.

Λόγω της μεγαλύτερης δημιουργικότητας των προγραμματιστών Flutter, απαιτείται λιγότερος χρόνος για τη δημιουργία μιας εφαρμογής, πράγμα που σημαίνει ότι σε σύγκριση με άλλες γλώσσες προγραμματισμού και πλαίσια, οι εφαρμογές στο Flutter γράφονται πιο γρήγορα και εισέρχονται στην αγορά νωρίτερα με την ίδια προσπάθεια. Έτσι, όσο λιγότερη χρειάζεται κωδικοποίηση και υποστήριξη, τόσο γρηγορότερος είναι ο χρόνος αγοράς.

Το Flutter παρέχει πιο αποτελεσματικές εργασίες ανάπτυξης και, κατά συνέπεια, η ανάπτυξη μιας εφαρμογής απαιτεί λιγότερες ανθρωποώρες. Ταυτόχρονα, το κόστος μιας ώρας είναι στο μέσο επίπεδο αγοράς. Ως αποτέλεσμα, το κόστος της εφαρμογής στο Flutter είναι χαμηλότερο από ό,τι όταν χρησιμοποιούνται άλλες γλώσσες μεταξύ πλατφορμών ή native ανάπτυξη.

3.4 Python

Η Python είναι μια γλώσσα γενικής χρήσης κάτι που σημαίνει ότι σε αντίθεση με HTML, CSS και JavaScript, μπορεί να χρησιμοποιηθεί για σε άλλους τύπους προγραμματισμού και ανάπτυξης λογισμικού εκτός από την ανάπτυξη διαδικτυακών εφαρμογών.

Η Python είναι γενικής χρήσης, ευέλικτη και δημοφιλής γλώσσα προγραμματισμού. Είναι υπέροχη ως πρώτη γλώσσα εκμάθησης, επειδή είναι περιεκτική και εύκολη στην ανάγνωση και είναι επίσης καλή γλώσσα να γνωρίζει κάθε προγραμματιστής, καθώς μπορεί να χρησιμοποιηθεί για τα πάντα, από την ανάπτυξη ιστού έως την ανάπτυξη λογισμικού και επιστημονικές εφαρμογές.

3.4.1 Χρήσεις Python

Εφαρμογές Διαδικτύου

Η Python διαθέτει προεγκατεστημένες βιβλιοθήκες και frameworks για Web Εφαρμογές, όπως Django και Flask, είναι ιδιαίτερα εξαιρετική για ανάπτυξη front-end και back-end έργων. Επίσης διαθέτει πολλές βιβλιοθήκες για εύκολη διαχείριση API και Sockets.

Οι διάφοροι τομείς χρήσης της Python δίνονται παρακάτω.

Επιστημονικά δεδομένα

Εξόρυξης

Εφαρμογές για υπολογιστή

Εφαρμογές με βάση την κονσόλα

Εφαρμογές για κινητά

Ανάπτυξη λογισμικού

Τεχνητή νοημοσύνη

Εφαρμογές Ιστού

Επιχειρηματικές εφαρμογές

Εφαρμογές 3D CAD

Μηχανική εκμάθηση

Εφαρμογές επεξεργασίας εικόνας

Αναγνώριση ομιλίας

3.4.2 Γιατί χρησιμοποιήθηκε η Python

Η python χρησιμοποιήθηκε στην εργασία λόγω πολλών βιβλιοθηκών για την χρήση των ακροδεκτών και των πολλών λειτουργιών του Raspberry. Επιπλέον, λόγω ευκολίας στην χρήση της βιβλιοθήκης requests και Flask είναι πολύ εύκολη η χρήση των get και post για την επικοινωνία του κινητού με την εφαρμογή στο raspberry όπως και αποστολή και λήψη δεδομένων από το Thingspeak.

3.5 Thingspeak

Το Διαδίκτυο των πραγμάτων (συσκευές) (IoT) είναι ένα σύστημα «συνδεδεμένων πραγμάτων». Τα πράγματα περιλαμβάνουν γενικά ένα ενσωματωμένο λειτουργικό σύστημα και μια ικανότητα επικοινωνίας με το Διαδίκτυο ή με τα γειτονικά πράγματα. Ένα από τα βασικά στοιχεία ενός γενικού συστήματος IoT που γεφυρώνει τα διάφορα «πράγματα» είναι μια υπηρεσία IoT. Μια ενδιαφέρουσα επίπτωση από τα «πράγματα» που περιλαμβάνουν τα συστήματα IoT είναι ότι τα πράγματα από μόνα τους δεν μπορούν να κάνουν τίποτα. Θα πρέπει να έχουν τη δυνατότητα να συνδεθούν με άλλα «πράγματα». Όμως, η πραγματική δύναμη του IoT αξιοποιείται όταν τα πράγματα συνδέονται με μια «υπηρεσία» είτε άμεσα είτε μέσω άλλων «πραγμάτων». Σε τέτοια συστήματα, η υπηρεσία παίζει το ρόλο ενός αόρατου διαχειριστή παρέχοντας δυνατότητες που κυμαίνονται από απλή συλλογή δεδομένων και παρακολούθηση έως πολύπλοκες αναλύσεις δεδομένων. Το παρακάτω διάγραμμα δείχνει πού μια υπηρεσία IoT ταιριάζει σε ένα οικοσύστημα IoT:

Μια τέτοια πλατφόρμα εφαρμογών IoT που μπορεί να προσφέρει μια ευρεία ποικιλία δυνατοτήτων ανάλυσης, παρακολούθησης και αντιδράσεων είναι το «ThingSpeak».

Τι είναι το ThingSpeak

Το ThingSpeak είναι μια πλατφόρμα που παρέχει διάφορες υπηρεσίες που στοχεύουν αποκλειστικά στην κατασκευή εφαρμογών IoT. Προσφέρει τις δυνατότητες συλλογής δεδομένων σε πραγματικό χρόνο, οπτικοποιώντας τα συλλεγόμενα δεδομένα με τη μορφή γραφημάτων, δυνατότητα δημιουργίας προσθηκών και εφαρμογών για συνεργασία με υπηρεσίες ιστού, κοινωνικό δίκτυο και άλλα API.

Το βασικό στοιχείο του ThingSpeak είναι ένα «ThingSpeak Channel». Ένα κανάλι αποθηκεύει τα δεδομένα που στέλνουμε στο ThingSpeak και περιλαμβάνει τα παρακάτω στοιχεία:

8 πεδία για την αποθήκευση δεδομένων οποιουδήποτε τύπου - Αυτά μπορούν να χρησιμοποιηθούν για την αποθήκευση των δεδομένων από έναν αισθητήρα ή από μια ενσωματωμένη συσκευή.

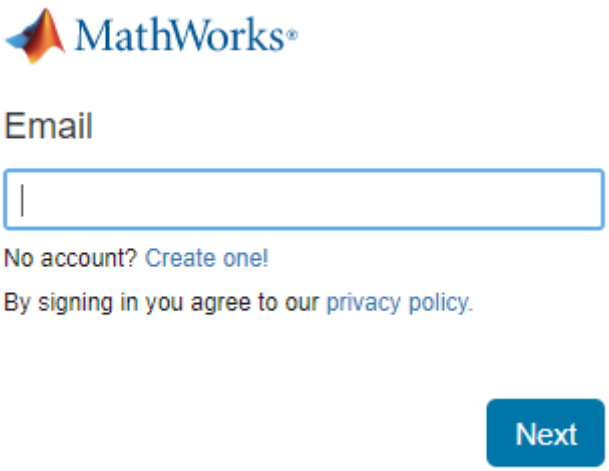
3 πεδία τοποθεσίας - Μπορεί να χρησιμοποιηθεί για την αποθήκευση του πλάτους, του μήκους και του υψομέτρου. Είναι πολύ χρήσιμα για την παρακολούθηση μιας κινητής συσκευής.

1 πεδίο κατάστασης - Ένα σύντομο μήνυμα για την περιγραφή των δεδομένων που είναι αποθηκευμένα στο κανάλι.

Για να χρησιμοποιήσετε το ThingSpeak, πρέπει να εγγραφείτε και να δημιουργήσετε ένα κανάλι. Μόλις έχουμε ένα κανάλι, μπορούμε να στείλουμε τα δεδομένα, να επιτρέψουμε στο ThingSpeak να το επεξεργαστεί και επίσης να ανακτήσει τα ίδια.

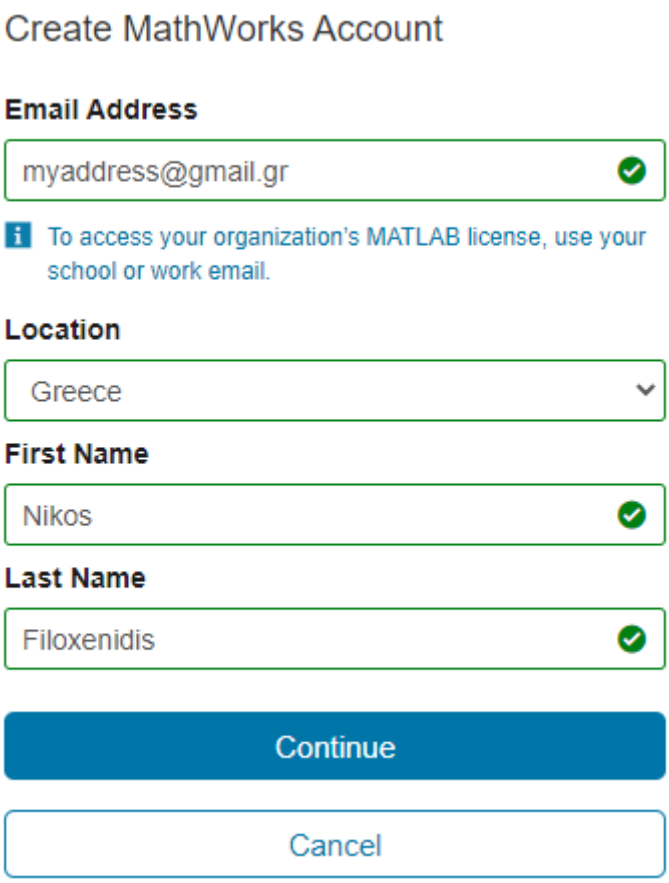
Ξεκινώντας

Ανοίξτε το https://thingspeak.com/ και κάντε κλικ στο κουμπί «Start» στο κέντρο της σελίδας και θα μεταφερθείτε στη σελίδα εγγραφής. Συμπληρώστε τις απαιτούμενες λεπτομέρειες και κάντε κλικ στο κουμπί «Δημιουργία λογαριασμού».



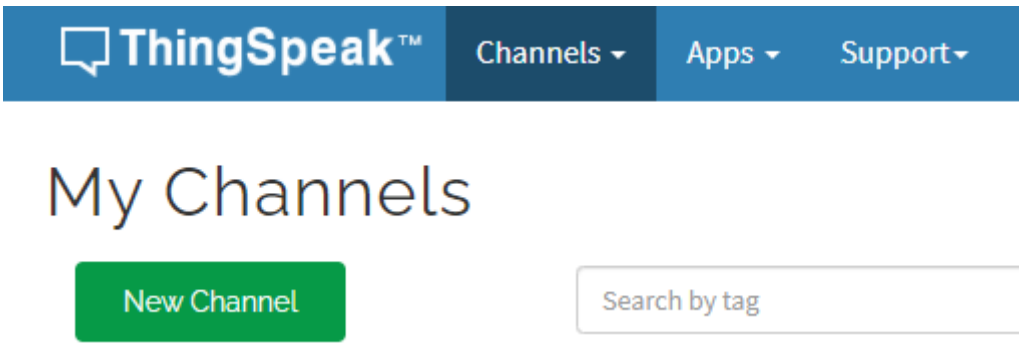
Εικόνα 3.6: Thingspeak – Δημιουργία λογαριασμού

Στην συνέχεια γράφετε τα στοιχεία σας και πατάτε το Continue-Συνέχεια, όπως φαίνεται στην εικόνα 3.6

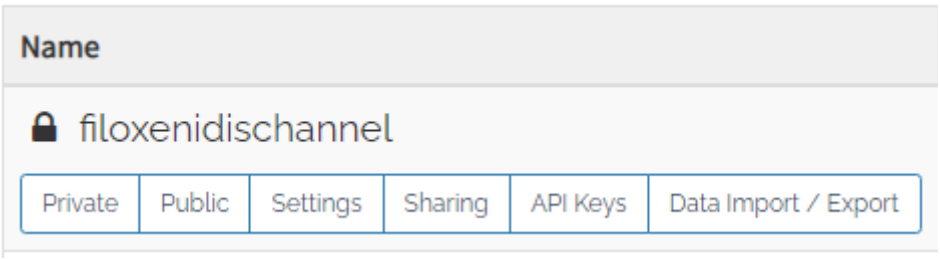


Εικόνα 3.7: Thingspeak – Δημιουργία λογαριασμού - Στοιχεία

Στη συνέχεια μπορείτε να δημιουργήσετε ένα νέο κανάλι

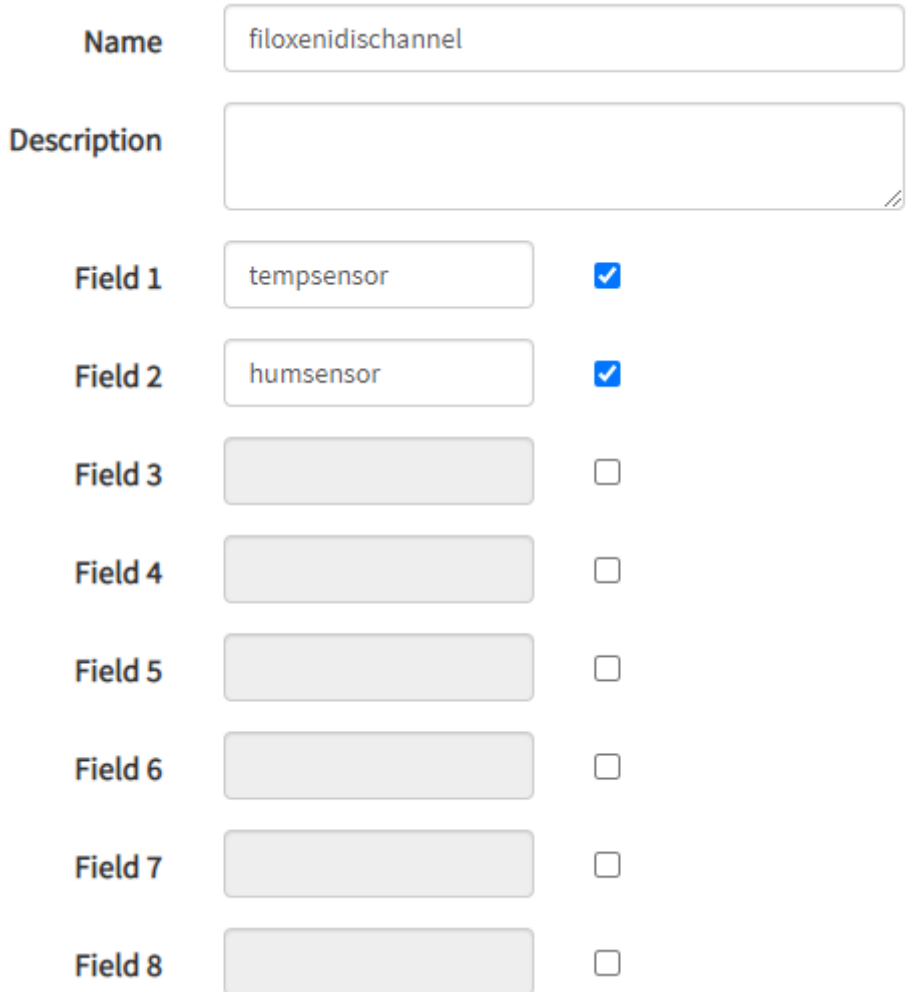


Εικόνα 3.8: Thingspeak – Δημιουργία Καναλιού

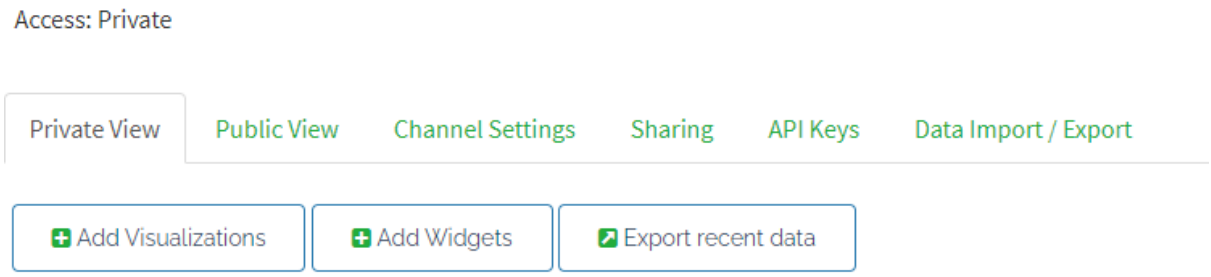


Εικόνα 3.9: Thingspeak – Ρυθμίσεις για το κανάλι

Μπορείτε να βάλετε μέχρι 8 πεδία, δηλαδή 8 αισθητήρες.

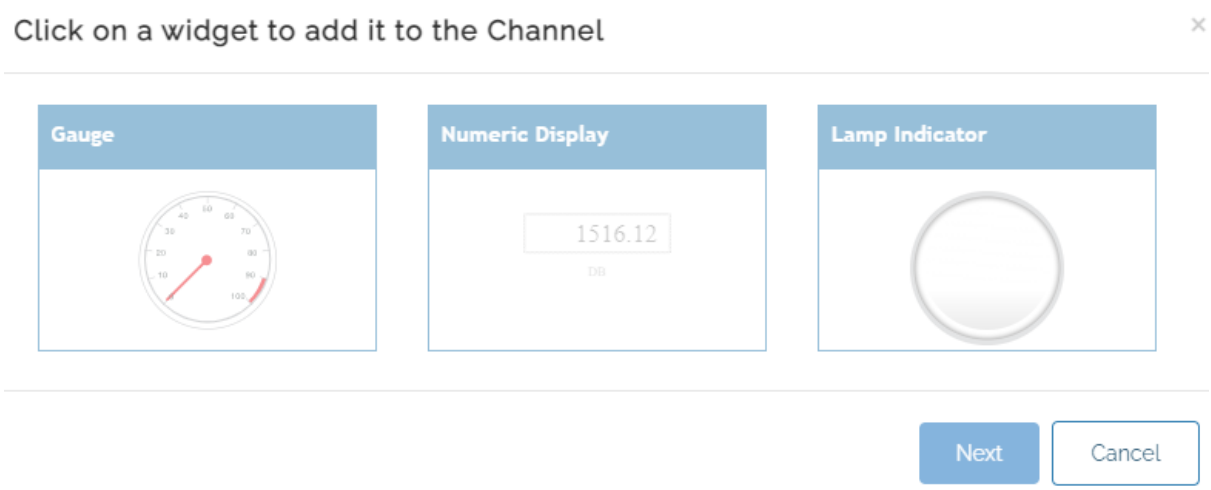


Εικόνα 3.10: Thingspeak – Χαρακτηριστικά Καναλιού- Πεδία

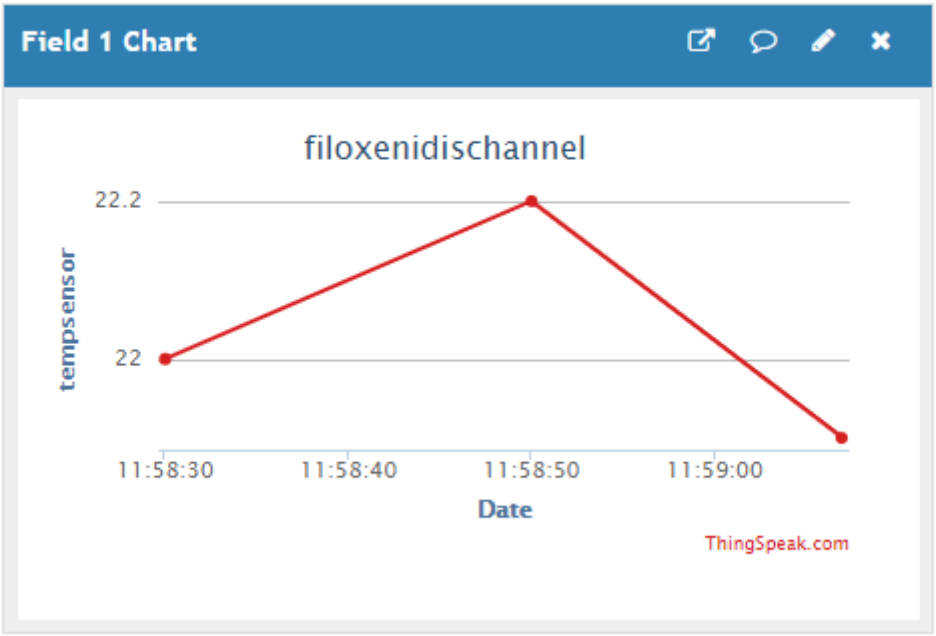


Εικόνα 3.11: Thingspeak – Προσθήκη πρόσθετων

Μπορείτε να βάλετε διάφορα πρόσθετα για να αναπαραστήσετε τις τιμές των πεδίων όπως φαίνεται στις εικόνες 3.11 και 3.12.

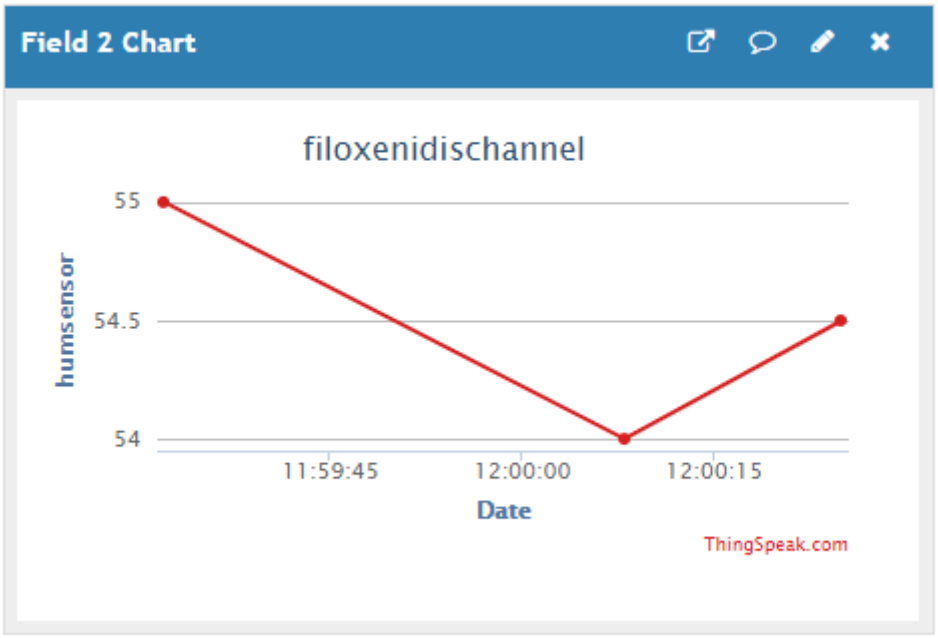


Εικόνα 3.12: Thingspeak – Πρόσθετα

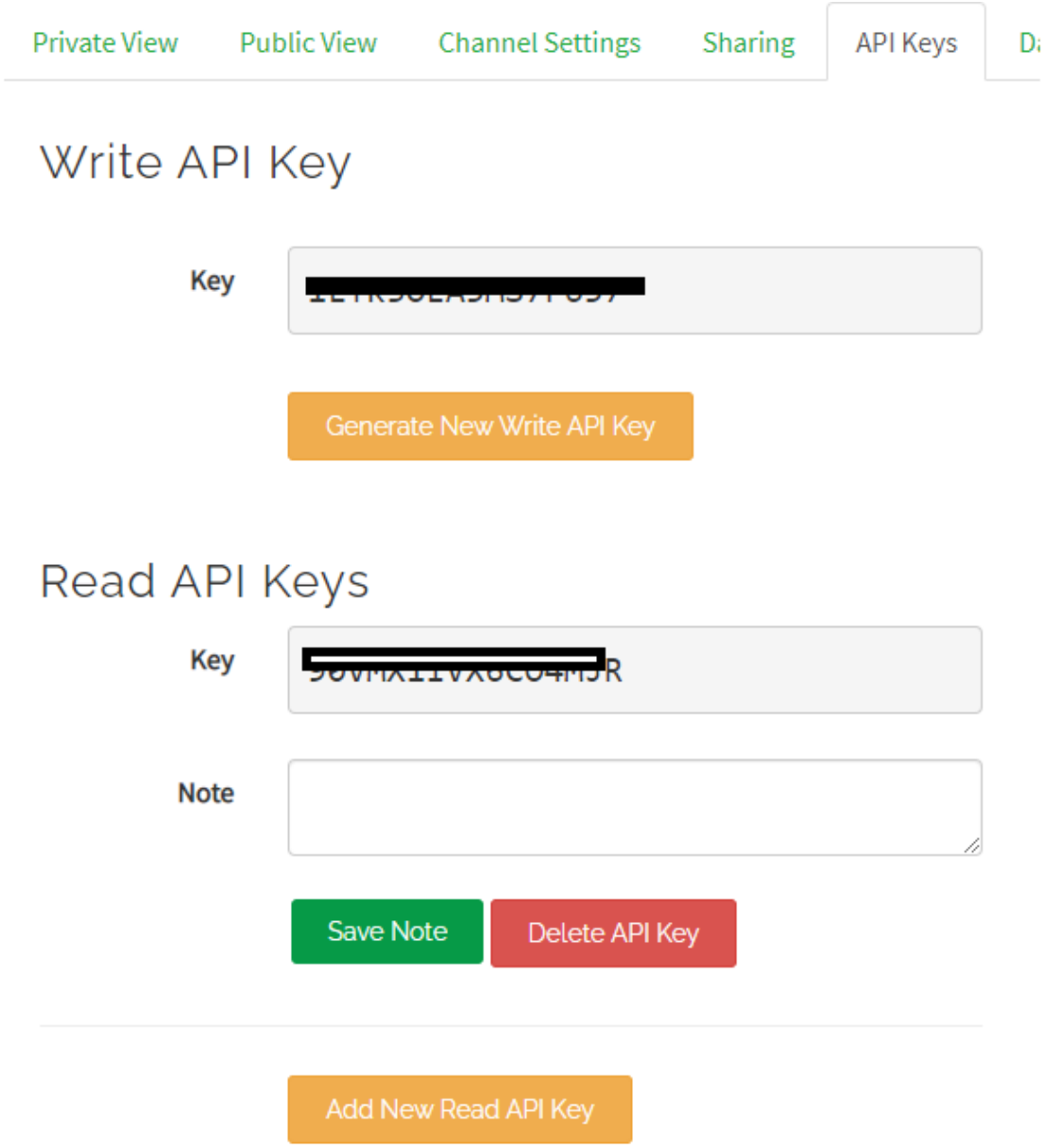


Εικόνα 3.13: Thingspeak – Εμφάνιση τιμών για το πεδίο Θερμοκρασίας

Στις Εικόνες 3.13 και 3.14 παρουσιάζονται τα γραφήματα των τιμών των αισθητήρων σε συνάρτηση με τον χρόνο.



Εικόνα 3.14: Thingspeak – Εμφάνιση τιμών για το πεδίο Υγρασίας



Εικόνα 3.15: Thingspeak – Τα κλειδιά του API για εγγραφή και ανάγνωση πεδίων από το κανάλι

Στην Εικόνα 3.15 παρουσιάζεται η καρτέλα που πρέπει να μπει ο ιδιοκτήτης ενός καναλιού και να ρυθμίσει τα κλειδιά που χρειάζονται για να κάνει αποστολή και λήψη των τιμών ενός πεδίου του καναλιού.

Ο κώδικας για αποστολή και λήψη παρουσιάζεται παρακάτω:

Write a Channel Feed

```
GET https://api.thingspeak.com/update?api_key=aaaaaaaaaaaa&field1=0
```

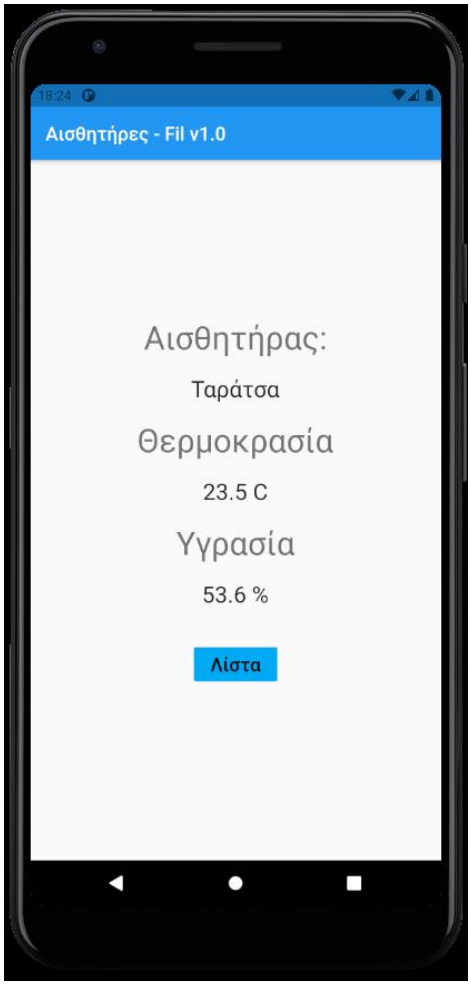
Read a Channel Field

```
GET https://api.thingspeak.com/channels/123456/fields/1.json?api_key=bbbbbbbbb&results=2
```

Κεφάλαιο 4ο: Το σύστημα

4.1 Εισαγωγή - Πλοήγηση

Στο κεφάλαιο αυτό θα παρουσιαστεί το σύστημα που υλοποιήθηκε όσον αφορά την πλοήγηση και τις λειτουργίες μέσω του κινητού-εφαρμογής και εξήγηση ορισμένων μερών του κώδικα με τα διαγράμματα τους.



Εικόνα 4.1: Το σύστημα – η εφαρμογή – Πρώτη (κεντρική) Οθόνη

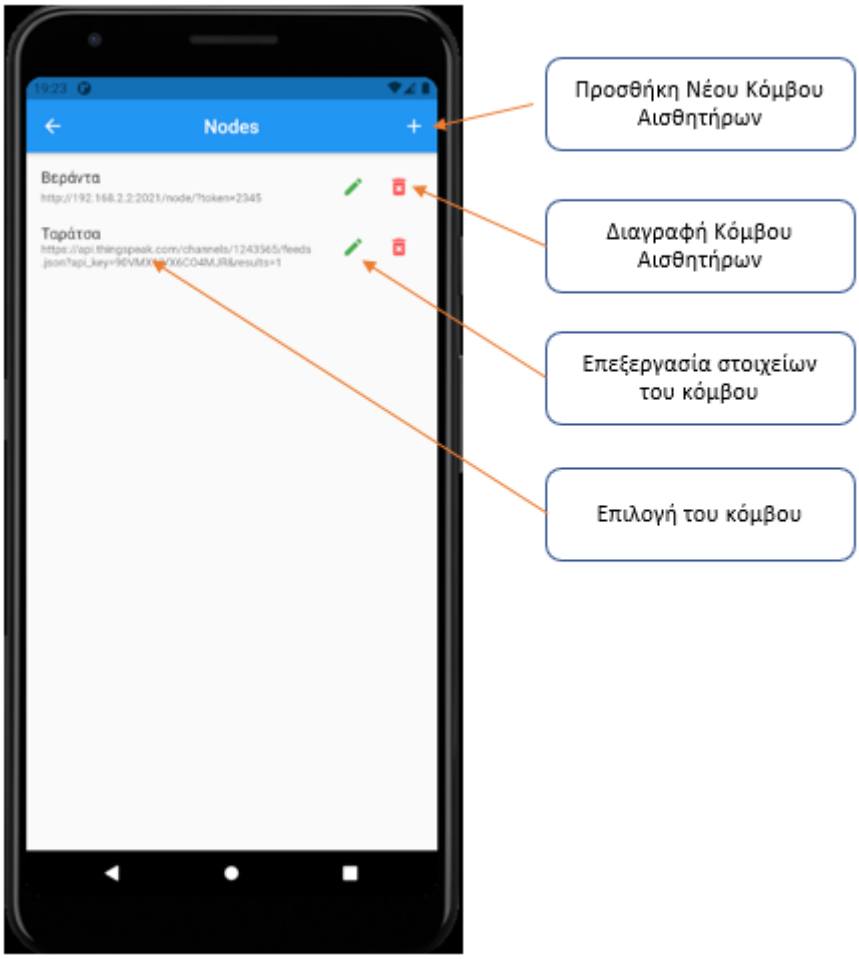
Στην εικόνα 4.1 παρουσιάζεται η πρώτη οθόνη στην εφαρμογή του κινητού όταν ο χρήστης την ανοίγει. Αν βρει κάποιον αισθητήρα-κόμβο που υπάρχει αποθηκευμένος τότε δείχνει τις τιμές. Στη οθόνη αυτή φαίνεται το όνομα που έχει δοθεί στον κόμβο και οι τιμές θερμοκρασίας και υγρασίας για τον συγκεκριμένο αισθητήρα.

Εικόνα 4.2: Οθόνη για επιλογή κόμβου

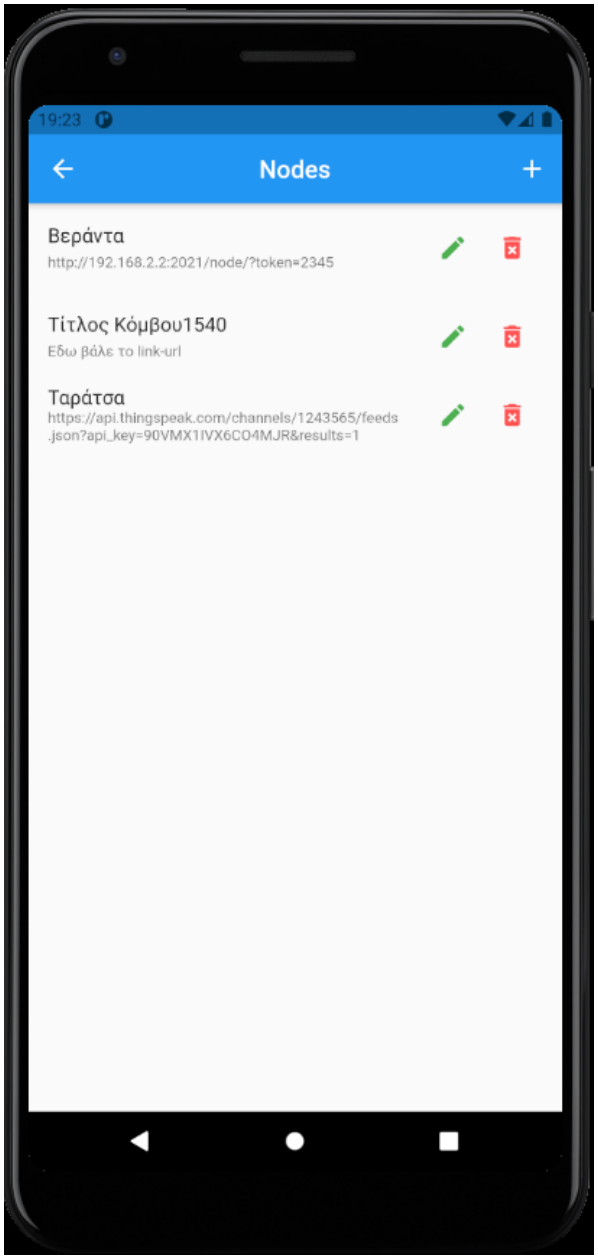
Στις εικόνες 4.2 και 4.3 παρουσιάζονται οι οθόνες όταν ο χρήστης στην οθόνη της εικόνας 4.1 πατήσει το κουμπί Λίστα. Σε αυτή μπορεί να επιλέξει κάποιον αισθητήρα-κόμβο που υπάρχει αποθηκευμένος.

Ο χρήστης έχει τις επιλογές για:

- Δημιουργία νέας καταχώρησης κόμβου-αισθητήρα
- Διαγραφή αποθηκευμένου κόμβου
- Επεξεργασία αποθηκευμένου κόμβου
- Επιλογή αποθηκευμένου κόμβου για προβολή τιμών στην πρώτη κεντρική



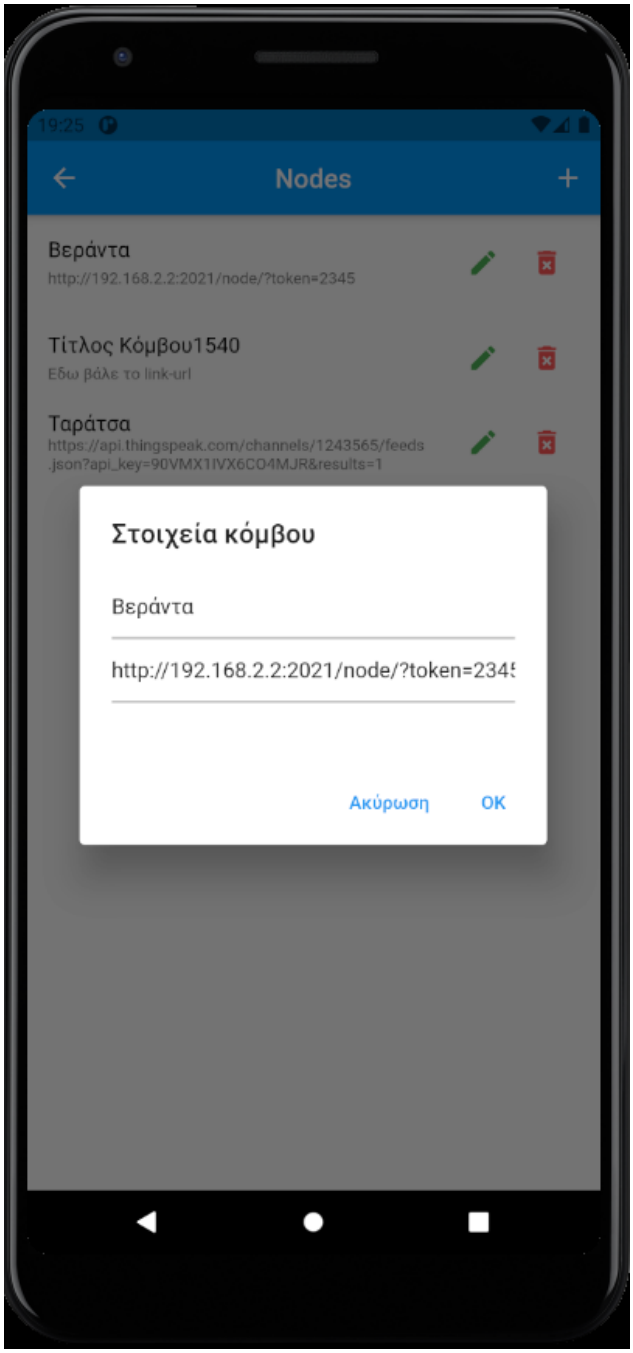
Εικόνα 4.3: Οθόνη για τις δυνατότητες για τη διαχείριση κόμβων



Εικόνα 4.4: Δημιουργία νέας καταχώρησης κόμβου

Στην εικόνα 4.4 παρουσιάζεται η οθόνη όταν ο χρήστης πατήσει το σταυρό για προσθήκη νέου κόμβου.

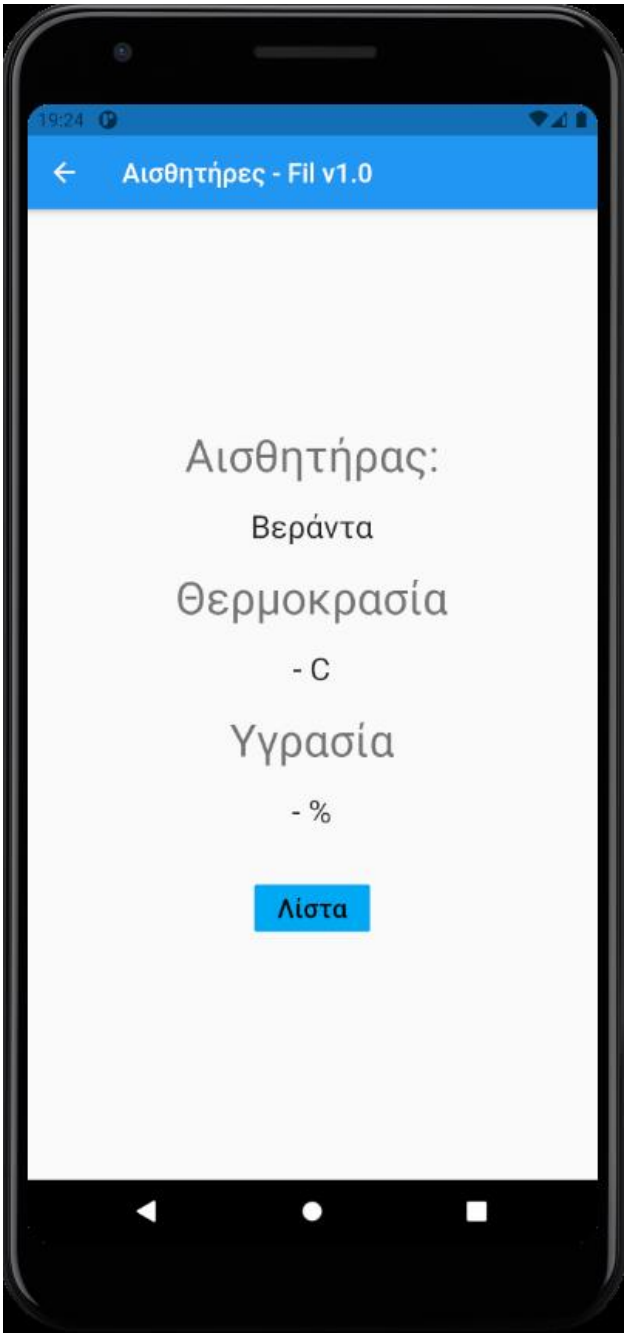
Τότε δημιουργείται νέος κόμβος με ένα τυχαίο όνομα και ο χρήστης μπορεί να αλλάξει το όνομα και να βάλει το url του κόμβου.



Εικόνα 4.5: Επεξεργασία στοιχείων κόμβου

Στην εικόνα 4.5 παρουσιάζεται η οθόνη όταν ο χρήστης έχει επιλέξει έναν κόμβο και προσπαθεί να τροποποιήσει τα στοιχεία του κόμβου, όπως

- Τίτλο
- Διεύθυνση



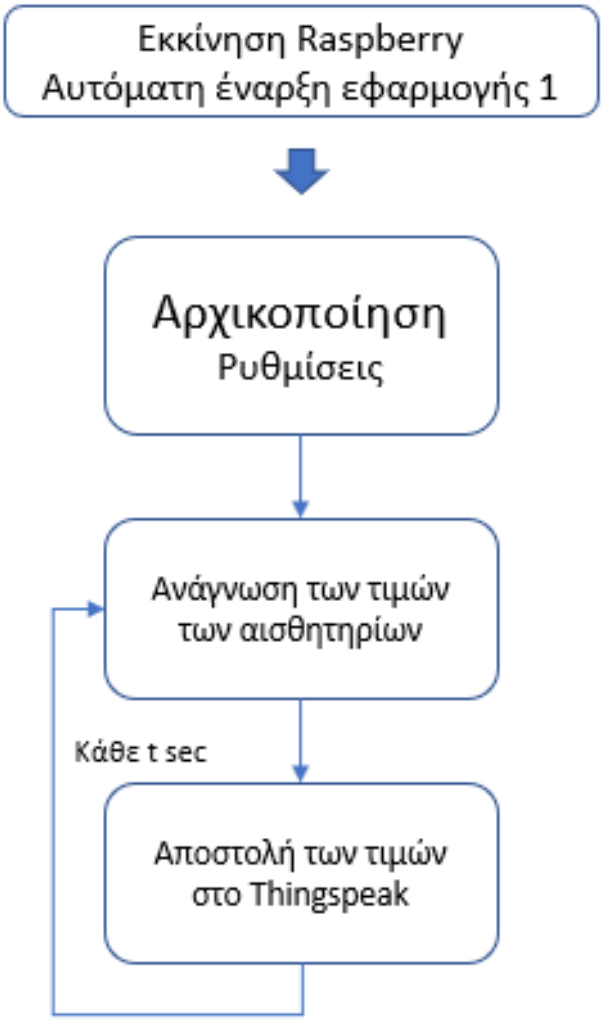
Εικόνα 4.6: Οθόνη απεικόνισης τιμών από τον αισθητήρα όταν παρουσιάζεται πρόβλημα

Στην εικόνα 4.6 παρουσιάζεται η οθόνη όταν ο χρήστης έχει επιλέξει έναν κόμβο και προσπαθεί να γίνει λήψη των τιμών αλλά οι τιμές δεν εμφανίζονται για διάφορους λόγους, όπως είναι λάθος στο url του κόμβου ή ο κόμβος δεν ανταποκρίνεται.

4.2 Επεξήγηση και διαγράμματα των εφαρμογών

4.2.1 Εφαρμογή στο Raspberry

Στο raspberry έχουν δημιουργηθεί δημιουργηθεί δύο κύρια προγράμματα σε Python για την εξυπηρέτηση τριών βασικών λειτουργιών, η επικοινωνία και η ανάγνωση τιμών από το αισθητήριο, η λήψη εντολής από την εφαρμογή από το κινητό για την αποστολή των δεδομένων σε αυτό και τέλος η αποστολή των δεδομένων του αισθητήρα στο Thingspeak.



Εικόνα 4.7: Raspberry – Εφαρμογή 1

Στην εικόνα 4.7 παρουσιάζεται το διάγραμμα λειτουργίας για την πρώτη εφαρμογή. Στην αρχή γίνεται αρχικοποίηση και στη συνέχεια γίνεται ανάγνωση των τιμών των αισθητηρίων και αποστολή των τιμών στο Thingspeak ανα χρονικό διάστημα που ορίζεται από τον προγραμματιστή.

Στη συνέχεια παρουσιάζεται ο κώδικας που υλοποιεί την παραπάνω διαδικασία.

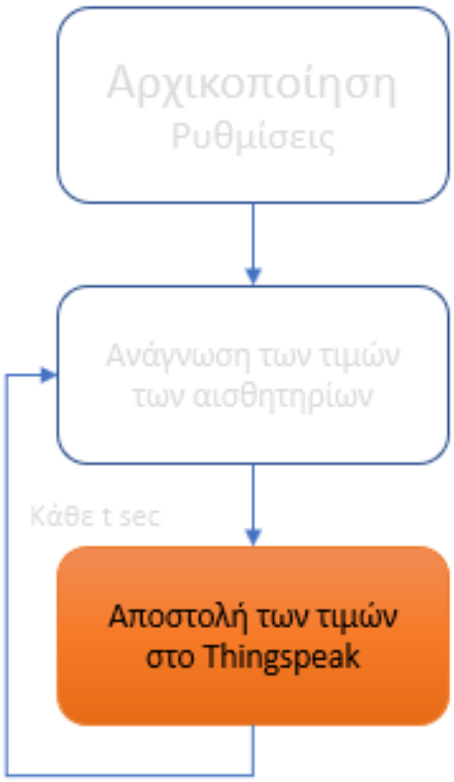
Χρειάζονται οι δύο σημαντικές βιβλιοθήκες Flask και request για την εγκατάσταση server και την επικοινωνία της εφαρμογής με το διαδίκτυο.

```
from flask import Flask
from flask import request,jsonify
import json
import requests


app = Flask(__name__)
```

Δημιουργείται μια συνάρτηση με route path / για να εξυπηρετήσει τα requests που έρχονται στην προεπιλεγμένη διεύθυνση.

```
@app.route('/', methods=['GET', 'POST'])
def main():
    return "Main"
```



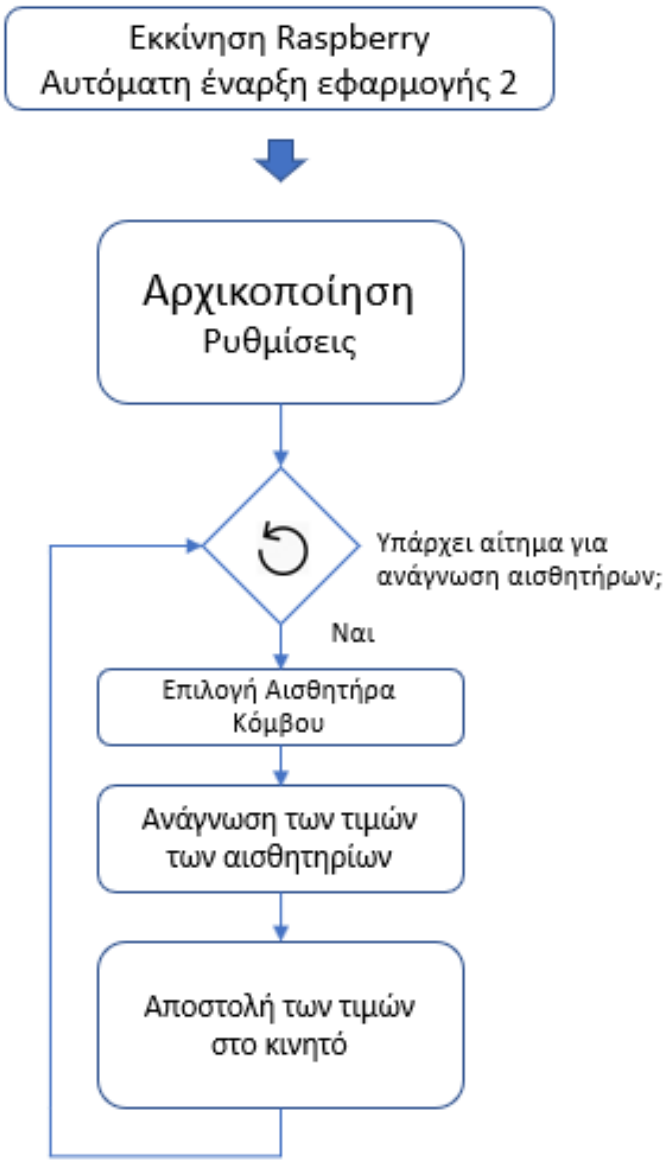
Εικόνα 4.8: Raspberry – Εφαρμογή 1 για το Thingspeak

Για να σταλθούν τιμές στο ThingSpeak πρέπει να στα καλεστεί το ένα url σε συγκεκριμένη μορφή όπως το παρακάτω

```
https://api.thingspeak.com/update.json?api_key=KEY&field1=23&field2=55
```

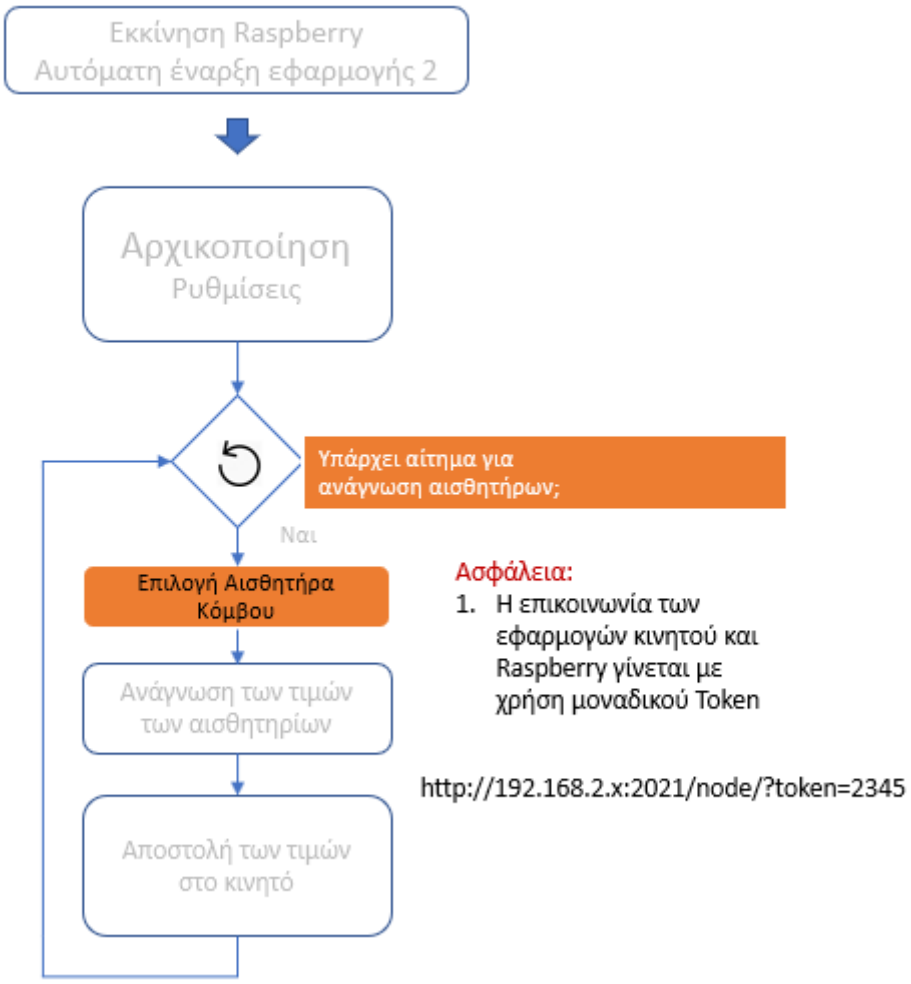
Δημιουργήθηκε το (συνάρτηση) route /node για το την αποστολή των δεδομένων του αισθητήρα στο thingspeak. Ενδεικτικά στο παράδειγμα αποστέλλονται 23 και 55 για θερμοκρασία και υγρασία.

```
@app.route('/node/', methods=['GET'])
def node():
gs1 = 23;
gs2 = 55;
requests.get("https://api.thingspeak.com/update.json?api_key=IAMKEYFORGET&field1="+str(gs1)
+"&field2="+str(gs2))
```



Εικόνα 4.9: Raspberry – Εφαρμογή 2 για το κινητό

Στην εικόνα 4.9 παρουσιάζεται το διάγραμμα λειτουργίας για τη δεύτερη εφαρμογή. Στην αρχή γίνεται αρχικοποίηση και στη συνέχεια γίνεται ανάγνωση των τιμών των αισθητηρίων που επιλέχθηκε από την εφαρμογή στο κινητό και αποστολή των τιμών πίσω σε αυτό ανα χρονικό διάστημα που ορίζεται από τον προγραμματιστή.



Εικόνα 4.10: Raspberry – Εφαρμογή 2 - Token

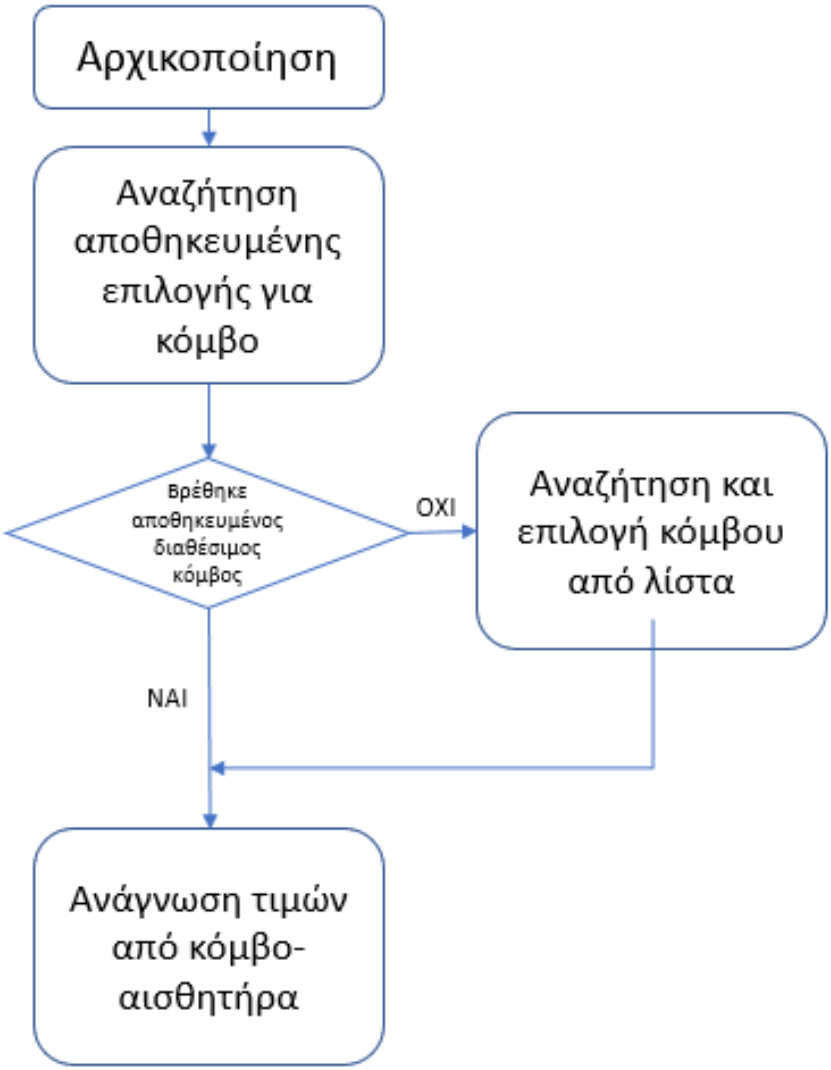
Στην εικόνα 4.10 παρουσιάζεται το διάγραμμα λειτουργίας για τη επιλογή αισθητήρα και τον τρόπο επικοινωνίας εφαρμογής στο κινητό και raspberry με χρήση token.

Για παράδειγμα όταν μια εφαρμογή στο κινητό θέλει να συνδεθεί σε ένα κόμβο πρέπει να κάνει αναζήτηση σε συγκεκριμένο url που έχει τοποθετηθεί από τον προγραμματιστή και την γνωρίζει ο χρήστης όπως παρουσιάζεται στην εικόνα 4.11. Στο url που τοποθετηθεί πρέπει να γράψει και ένα μοναδικό token που είναι χαρακτηριστικό-μοναδικό και ορισμένο στον κώδικα στο raspberry

Στον παρακάτω κώδικα παρουσιάζεται ο τρόπος με τον οποίο ελέγχεται στο raspberry το token. Δεν μπορεί να εξυπηρετηθεί η εφαρμογή στο κινητό από το raspberry να δεν γνωρίζει το token.

```
@app.route('/node/', methods=['GET'])
def node():
    token = request.args.get("token");
```

```
if token == '2345':
    sensors = {"temp": 23, "hum": 56}
    return json.dumps(sensors)
else:
    return "Not Authorized"
```



Εικόνα 4.11: Εφαρμογή στο κινητό - Διάγραμμα

4.3 Η ασφάλεια στο σύστημα διαχείρισης των δεδομένων και επικοινωνίας

Ασφάλεια χρήσης της εφαρμογής στο κινητό

Η εφαρμογή έχει δημιουργηθεί με Flutter για Android κινητό. Οποιοσδήποτε έχει την εφαρμογή μπορεί να βάλει το url ενός κόμβου ή του Thingspeak.

Ασφάλεια πρόσβασης σε κόμβο

Οποιοσδήποτε μπορεί να έχει πρόσβαση στο τοπικό δίκτυο μπορεί να βάλει το url ενός κόμβου και να ζητήσει να του επιστραφούν οι τιμές από τους αισθητήρες. Όμως δεν θα καταφέρει να λάβει σωστή απάντηση γιατί λόγω του token. Κάθε κόμβος μπορεί να έχει μοναδικό (ή και περισσότερα) token κλειδιά όπως περιεγράφηκε σε προηγούμενη υποενότητα. Όταν ο χρήστης τοποθετεί το url του κόμβου στην εφαρμογή πρέπει να βάζει και το token του κόμβου.

Με αυτόν τον τρόπο διασφαλίζεται ότι κανένας μη εξουσιοδοτημένος δεν μπορεί να έχει πρόσβαση στον κόμβο όταν είναι διασυνδεμένος στο τοπικό δίκτυο.

Όταν ένας χρήστης δεν είναι συνδεδεμένος στο ίδιο τοπικό δίκτυο θα πρέπει να χρησιμοποιήσει την υπηρεσία Thingspeak. Θα πρέπει να καλέσει το κατάλληλο url της υπηρεσίας και να διαβάσει τις τιμές από τον κόμβο που τον ενδιαφέρει. Όπως αναλύθηκε σε προηγούμενη υποενότητα ο χρήστης πρέπει να γνωρίζει το κανάλι και το κλειδί για λήψη για να έχει πρόσβαση.

Με αυτό τον τρόπο διασφαλίζεται ότι κανένας δεν μπορεί να λάβει τις τιμές από έναν κόμβο αν δεν γνωρίζει τα παραπάνω στοιχεία.

Όπως αναφέρθηκε κάθε κόμβος αποστέλλει σε συγκεκριμένο χρονικό διάστημα τα δεδομένα από τους αισθητήρες του στο Thingspeak. Αυτό γίνεται μέσω ειδικών κλειδιών και με αυτό τον τρόπο διασφαλίζεται ότι κανένας που δεν γνωρίζει αυτά τα κλειδιά δεν μπορεί να τοποθετήσει τιμές στο Thingspeak.

Τέλος να τονισθεί ότι η επικοινωνία σε τοπικό επίπεδο δεν χρησιμοποιεί SSL ενώ κάθε επικοινωνία με το Thingspeak χρησιμοποιεί SSL σε κάθε λειτουργία.

Κεφάλαιο 5ο: Συμπεράσματα και προτάσεις βελτίωσης

Παρουσιάστηκε και αναλύθηκε ένα σύστημα παρακολούθησης αισθητήρων σε ένα χώρο μέσω τοπικού δικτύου ή μέσω διαδικτύου. Υλοποιήθηκε android εφαρμογή παρακολούθησης αισθητήρων που είναι συνδεδεμένα σε raspberry pi 3. Η εφαρμογή android επικοινωνεί με το raspberry μέσω wifi για την παρακολούθηση θερμοκρασίας και υγρασίας σε έναν χώρο. Ο χρήστης μπορεί μέσω της φορητής συσκευής μικρού κόστους και κατανάλωσης να επιβλέπει με το κινητό του τις περιβαντολλογικές συνθήκες. Ένα από τα πιο σημαντικά χαρακτηριστικά του συστήματος είναι η δυνατότητα προσθήκης περισσότερων κόμβων που μπορεί να διαχειριστεί η εφαρμογή. Επίσης, είναι η επίβλεψη των τιμών των αισθητήρων διαδικτυακά μέσω της υπηρεσίας Thingspeak. Κάθε κόμβος μπορεί να στέλνει τα δεδομένα του μέσω ασφαλούς καναλιού στο Thingspeak και ο χρήστης μέσω της εφαρμογής να τα επιβλέπει από το κινητό του.

Επιπλέον παρέχεται η δυνατότητα προσθήκης, επεξεργασίας και διαγραφής νέων κόμβων μέσω της εφαρμογής.

Η ασφάλεια του συστήματος περιεγράφηκε στο προηγούμενο κεφάλαιο και μπορούμε να συμπεράνουμε ότι η χρήση token σε κάθε requests που γίνεται τοπικά διασφαλίζει την αυθεντικοποίηση του χρήστη.

Όσον αφορά τις βελτιώσεις από τις πιο σημαντικές είναι η προσθήκη username και password στην εφαρμογή για να μην μπορεί κάποιος να ανοίξει την εφαρμογή. Επίσης, είναι αποφυγή του Port forwading ώστε η παρακολούθηση των τιμών να γίνει μέσω διαδικτύου χωρίς να απαιτείται η χρήση της υπηρεσίας Thingspeak.

ΒΙΒΛΙΟΓΡΑΦΙΑ

- "Raspberry Pi Zero: the $5 Computer". Raspberry Pi Foundation. November 2015
- "Features Raspberry Pi Today”. July 2015
- "Ten millionth Raspberry Pi, and a new kit – Raspberry Pi". Raspberry Pi. September 2016
- "Raspberry Pi 3 Model B+ on Sale at $35". Raspberry Pi Blog. Raspberry Pi Foundation. May 2018
- "Human Interface Guidelines". developer.apple.com. 2019.
- "Introduction to widgets". flutter.dev. 2020.
- "Material Design Widgets - Flutter". flutter.dev. 2017.
- "Flutter SDK releases". April 2021.

ΠΑΡΑΡΤΗΜΑ Α

Στο παράρτημα αυτό αναφέρονται τα βασικά κομμάτια του κώδικα που χρησιμοποιήθηκε.

Flutter - Nodeslist.dart

```
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:http/http.dart' as http;
import 'dart:convert' as convert;
import 'dart:math';
import 'package:line_icons/line_icons.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:appsensorsiot/node.dart';
import 'package:appsensorsiot/main.dart';

class NodesListScreen extends StatefulWidget {
  @override
  _NodesListScreenState createState() => _NodesListScreenState();
}

class _NodesListScreenState extends State<NodesListScreen> {
  Future save({Node nodes}) async {
    final pref = await SharedPreferences.getInstance();

    pref.setStringList(nodes.title, [nodes.title, nodes.url]).then((value) {
      value ? print('added') : print('not added');
    });
  }

  Future<List<Node>> getAllSavedData() async {
    final pref = await SharedPreferences.getInstance();

    var favs = pref.getKeys();
    List<String> keys = [];
    List<Node> listNodes = [];

    favs.forEach((element) {
      if (element.indexOf('selectednodetitle') == -1 &&
          element.indexOf('selectednodeurl') == -1) keys.add(element);
    });

    keys.forEach((element) {
      if (element != 'null') {
        List<String> content = pref.getStringList(element) ?? "_";
        Node nodes = Node(title: content[0], url: content[1]);
```

```
      listNodes.add(nodes);
    }
  });
  listNodes.sort((a, b) => a.title.compareTo(b.title));
  return listNodes;
}

final TextEditingController nodetitleController = TextEditingController();
final TextEditingController nodeurlController = TextEditingController();

showAlertDialog(BuildContext context, Node node) {
  nodetitleController.text = node.title;
  nodeurlController.text = node.url;
  String oldnodetitle = nodetitleController.text;

  AlertDialog alert = AlertDialog(
    title: Text('Στοιχεία κόμβου'),
    content: Column(
      mainAxisSize: MainAxisSize.min,
      children: <Widget>[
        TextField(
          controller: nodetitleController,
          keyboardType: TextInputType.numberWithOptions(decimal: true),
          decoration: InputDecoration(hintText: "Τίτλος"),
        ),
        TextField(
          controller: nodeurlController,
          keyboardType: TextInputType.numberWithOptions(decimal: true),
          decoration: InputDecoration(hintText: "Url-link"),
        ),
        SizedBox(
          height: 20,
        ),
      ],
    ),
    actions: <Widget>[
      TextButton(
        child: const Text('Ακύρωση'),
        onPressed: () {
          Navigator.of(context).pop();
        },
      ),
      TextButton(
          child: const Text('OK'),
          onPressed: () {

            //clear the old
```

```
              clearPerItem(title: oldnodetitle);
              //create the new
              Node node = Node(
                  title: nodetitleController.text, url: nodeurlController.text
);
              save(nodes: node);
              Scaffold.of(context).showSnackBar(SnackBar(
                  duration: Duration(milliseconds: 1500),
                  backgroundColor: Colors.blueGrey,
                  content: Text('Ανανεώθηκε')));
              setState(() {});

              Navigator.of(context).pop();
            }),
      ],
    );

    // show the dialog
    showDialog(
      context: context,
      builder: (BuildContext context) {
        return alert;
      },
    );
  }

  Future clearPerItem({String title}) async {
    final pref = await SharedPreferences.getInstance();

    pref.remove(title);
  }

  Future setSelectedNode(String title, String url) async {
    final pref = await SharedPreferences.getInstance();

    pref.setString("selectednodetitle", title);
    pref.setString("selectednodeurl", url);
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        actions: <Widget>[
          IconButton(
              icon: Icon(Icons.add),
              onPressed: () {
                var rng = new Random();
```

```
          Node nodes = Node(
              title: 'Τίτλος Κόμβου' + rng.nextInt(2000).toString(),
              url: 'Εδω βάλε το link-url');
          save(nodes: nodes);
          setState(() {});
        })
  ],
  centerTitle: true,
  title: Text(
    'Nodes',
    style: TextStyle(fontSize: 20),
  ),
),
body: FutureBuilder<List<Node>>(
    future: getAllSavedData(),
    builder: (context, data) {
      if (data.hasData) {
        return ListView.builder(
            itemCount: data.data.length,
            itemBuilder: (context, index) {
              return ListTile(
                onTap: () {
                  setSelectedNode(data.data[index].title.trim(),
                      data.data[index].url.trim());
                  Navigator.push(
                      context,
                      MaterialPageRoute(
                          builder: (context) => MyHomePage()));
                },
                title: Text(
                  data.data[index].title.trim(),
                  style: TextStyle(color: Colors.black87),
                ),
                subtitle: Text(
                  data.data[index].url.trim(),
                  style: TextStyle(fontSize: 12),
                ),
                trailing: Row(
                    mainAxisSize: MainAxisSize.min,
                    children: <Widget>[
                      IconButton(
                        onPressed: () {
                          showAlertDialog(context, data.data[index]);
                          setState(() {});
                        },
                        icon: Icon(
                          Icons.edit,
                          color: Colors.green,
```

```
                                ),
                              ),
                              IconButton(
                                onPressed: () {
                                  clearPerItem(title: data.data[index].title);
                                  Scaffold.of(context).showSnackBar(SnackBar(
                                      duration: Duration(milliseconds: 1500),
                                      backgroundColor: Colors.red,
                                      content: Text('Node has been Deleted')));
                                  setState(() {});
                                },
                                icon: Icon(
                                  Icons.delete_forever,
                                  color: Colors.redAccent,
                                ),
                              ),
                            ]),
                        );
                      });
                } else {
                  return Center(
                    child: Text(
                      "Empty",
                      style: TextStyle(color: Colors.black87),
                    ),
                  );
                }
              }),
        );
      }
    }
```

Τα πιο σημαντικά μέρη του κώδικα στο Raspberry

Εφαρμογή 1

```
import RPi.GPIO as GPIO
import Adafruit_DHT
import time
import requests

DHT_SENSOR = Adafruit_DHT.DHT11
DHT_PIN = 4
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)

while True:
    humidity, temperature = Adafruit_DHT.read(DHT_SENSOR, DHT_PIN)
    if humidity is not None and temperature is not None:

requests.get("https://api.thingspeak.com/update.json?api_key=1LYK3OEA5MS7F657&
field1="+str(humidity)+"&field2="+str(temperature))
        time.sleep(1)
```

Εφαρμογή 2

```
from flask import Flask
from flask import request,jsonify
import json
import requests
import RPi.GPIO as GPIO
import Adafruit_DHT

DHT_SENSOR = Adafruit_DHT.DHT11
DHT_PIN = 4
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)

app = Flask(__name__)

@app.route('/', methods=['GET', 'POST'])
def main():
    return "Main"

@app.route('/node/', methods=['GET'])
def node():
    token = request.args.get("token");
    if token == '2345':
        humidity, temperature = Adafruit_DHT.read(DHT_SENSOR, DHT_PIN)
        sensors = {"temp": humidity, "hum": temperature}
```

```
        return json.dumps(sensors)
    else:
        return "Not Authorized"

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=2021)
```