Πανεπιστήμιο Δυτικής Μακεδονίας
Τμήμα Ηλεκτρολόγων Μηχανικών Τ.Ε

Πτυχιακή Εργασία

Μετεωρολογικός Σταθμός με τη χρήση LoRa και WiFi.

Η παρούσα εργασία παρουσιάζεται για την λήψη του πτυχίου
Ηλεκτρολόγου Μηχανικού και Μηχανικού Υπολογιστών.

Συγραφέας: Κεραμιδιώτης Βασίλειος A.M: HN08293
Επιβλέπων: Βανδίκας Ιωάνης

Περίληψη

Στην παρούσα πτυχιακή εργασία με θέμα ¨Μετεωρολογικός Σταθμός με την χρήση WiFi-LoRa και Arduino' Θα αναλυθεί όλη η διαδικασία και τα βήματα που χρησιμοποιήθηκαν για την κατασκευή και τον προγραμματισμό του συστήματος αυτού.

Λέξεις Κλειδιά:

- Arduino,
- LoRa, LoRaWAN,
- The Things Network,
- github,
- WiFi,
- IoT,
- ThingSpeak,
- ESP8266

Περιεχόμενα

1 Εισαγωγή **5**

2 Ο Ρόλος ενός μετεωρολογικού σταθμού. **6**
2.1 Χρήση των δεδομένων 6

3 Θερμοκρασία **7**
3.1 Μέτρηση Θερμοκρασίας 8
3.2 Αισθητήρια Θερμοκρασίας 8

4 Ατμοσφαιρική Πίεση **9**
4.1 Μέτρηση Ατμοσφαιρικής Πίεσης 10
4.2 Αισθητήρια Πίεσης 10

5 Υγρασία **11**
5.1 Τύποι Υγρόμετρων 11

6 IoT(Internet of Things) **12**

7 Διασυνδέσεις **12**
7.1 LoRa 12
7.1.1 LoRa Διαμόρφωση 13
7.1.2 LoRaWAN 14
7.1.3 Αρχιτεκτονική LoRaWAN 14
7.2 WiFi 16
7.2.1 IEEE 802.11 16
7.2.2 Διαμόρφωση OFDM 16

8 Σχεδιασμός κυκλώματος **17**
8.1 Αισθητήρες Θεμοκρασίας Πίεσης και LCD 17
8.2 Αισθητήρας Σχετικής Υγρασίας 18
8.3 WiFi Module 18
8.4 LoRa Module 18
8.5 FBD Μετεωρολογικού σταθμού 19
8.6 Κυκλωματικό Διάγραμμα 20

9 Προγραμματισμός Σταθμού **21**
9.1 Προγραμματισμός Arduino 21
9.1.1 Κώδικας Flow Chart 21
9.1.2 Κώδικας Text 34

9.2 Προγραμματισμός ESP8266 50
9.2.1 Κώδικας Flow Chart 50
9.2.2 Κώδικας Text 57

10 Δημιουργία Εφαρμογής ThingSpeak **61**
10.1 Βήμα 1ο Δημιουργία Λογαριασμού 61
10.2 Βήμα 2ο Δημιουργία Καναλιού 62
10.3 Βήμα 3ο Συμπλήρωση Πεδίων 63
10.4 Βήμα 4ο Αντιγραφή Κλειδιών API 65
10.5 Βήμα 5ο Εισαγωγή Γραφημάτων και Οργάνων 68

11 Δημιουργία Εφαρμογής The Things Network **71**
11.1 Βήμα 1ο Δημιουργία λογαριασμού 71
11.2 Βήμα 2ο Δημιουργία εφαρμογής 73
11.3 Βήμα 3ο Καταχώριση συσκευής 76
11.4 OTAA (Over the Air Activation) 78

12 Συμπεράσματα και Παρατηρήσεις **81**
12.1 Έγκριση από ITU 81
12.2 Μετρήσεις και βελτιώσεις 82
12.3 Βελτιώσεις 82
12.4 Μετρήσεις 82

13 Φωτογραφία Σταθμού **84**

14 Βιβλιογραφία **85**

15 Παράρτημα **87**

1 Εισαγωγή

Ο Αναγνώστης αυτού του εγγράφου θα πρέπει να έχει βασικές γνώσεις προγραμματισμού, ηλεκτρονικών κυκλωμάτων και σημάτων.
Οι πηγές που χρησιμοποιήθηκαν για την δημιουργία αυτής της εργασίας αποτελούν κυρίως τεχνικά έγγραφα κατασκευαστών (datasheet), των ιστοσελίδων τους καθώς και πηγές για την γλώσσα προγραμματισμού του κυρίως προγράμματος και των δημιουργών των βιβλιοθηκών που χρησιμοποιήθηκαν στον κυρίως πρόγραμμα.

2 Ο Ρόλος ενός μετεωρολογικού σταθμού.

Οι Μετεωρολογικοί σταθμοί είναι συνήθως επίγεια κτήρια ή αυτοματισμοί, σε πολλά σημεία στην Ελλάδα αλλά και στον κόσμο τα οποία είναι εξοπλισμένα με μετρητικά όργανα και αναλυτές όπου η δουλειά τους είναι η συλλογή και ανάλυση καιρικών δεδομένων όπως:

- Θερμοκρασία
- Ταχύτητα ανέμου
- Διεύθυνση ανέμου
- Υγρασία
- Ατμοσφαιρική Πίεση
- Ηλιοφάνεια
- Βροχόπτωση
- Χιονόπτωση
- κ.α...

Τα οποία είναι μερικά, κυρίως βασικά δεδομένα που καταμετρά ένας σταθμός, φυσικά η λίστα το δεδομένων που πιθανός μπορεί να μετρήσει κάποιος είναι αρκετά μεγάλη.

2.1 Χρήση των δεδομένων

Οι λόγοι που μετράμε αυτά τα δεδομένα είναι διότι στην καθημερινότητα μας βασιζόμαστε σε αυτά. Από τον ποιο απλό λόγο όπως την μεταφορά μας και το ντύσιμο μας, μέχρι και την αποφυγή καταστροφών. Μερικοί λόγοι είναι από αυτούς είναι: Μέσα μεταφοράς(αυτοκίνητα,λεωφορεία,αεροσκάφη,πλοία), Πρωτογενής τομέας(γεωργία, κτηνοτροφία, κλπ.), βιομηχανία, κατασκευή, παραγωγή ηλεκτρικού ρεύματος, κ.α..

Το ποιο σημαντικό φυσικά είναι η πρόγνωση ακραίων καιρικών φαινομένων και φυσικών καταστροφών, για την λήψη μέτρων και την αποφυγή απώλειας ζωής και την απώλεια περιουσίας, των οποίων τα αποτελέσματα παρατηρούμε συχνά.

3 Θερμοκρασία

Το όργανο για την μέτρηση της Θερμοκρασίας είναι το Θερμόμετρο είτε αναλογικό είτε ψηφιακό, και μετράται σε βαθμούς κελσίου (oC), βαθμούς Φαρεναϊτ (F) και βαθμούς κέλβιν (K).

Σχήμα 1: αναλογικό θερμόμετρο οινοπνεύματος
[4]

Σχήμα 2: ψηφιακό θερμόμετρο
[4]

3.1 Μέτρηση Θερμοκρασίας

Ο Τρόπος που θα μετρήσουμε την θερμοκρασία είναι με ψηφιακό αισθητήριο τύπου IC (Ολοκληρομένου Κυκλώματος) το οποίο περιέχει μέσα στο πακέτο το στοιχείο της μέτρησης θερμοκρασίας καθώς και καταχωρητές δηλαδή μικρές θέσεις μνήμης στις οποίες μπορούμε να θέσουμε τιμές που μπορούν να μεταβάλουν παραμέτρους κατά την εκτέλεση της μέτρησης όπως τη διακριτική ικανότητα του, την απόκριση του κ.α. Φυσικά συνδεδεμένο με το IC είναι και ένα δεύτερο το οποίο είναι υπεύθυνο για την επικοινωνία με τον μικροελλεγκτή σύμφωνα με το πρωτόκολλο που που χρησιμοποιή π.χ (I2C,SPI,UART)κ.α.

3.2 Αισθητήρια Θερμοκρασίας

Ο τρόπος που χρησιμοποιούμε δεν είναι και ο μοναδικός. Άλλα αισθητήρια που χρησιμοποιούνται σε πολλές και διάφορες εφαρμογές είναι περιληπτικά:

- Τα θερμοζεύγη τα οποία βασίζοται στο φαινόμενο Seebeck το οποίο μας εξηγεί ότι όταν έχουμε διαφορά θερμοκρασίας μεταξύ δύο σημείων ενός ηλεκτρικά αγώγιμου υλικού, παρουσιάζεται ηλεκτρεργετική δύναμη μεταξύ τους την οποία μπορούμε να μετρήσουμε.
- Τα θερμόμετρα αντίστασης τα οποία παρουσιάζουν μεταβολή της αντίστασης τους ανάλογα με την θερμοκρασία και σύμφωνα με τιμή αντίστασης που παρουσιάζουν και τυπική τιμή του θερμικού συντελεστή στους 0oC που μας δίνονται από πίνακες μπορούμε να υπολογίσουμε την ακριβή θερμοκρασία.
- Τα θερμίστορ τα οποία είναι διάφορα είδη ημιαγωγών τα οποία παρουσιάζουν μη γραμμική μεταβολή της αντίστασης τους και μπορεί να είναι δύο τύπων αρνητικής μεταβολής (NTC), δηλαδή πτώση της αντίστασης με την άνοδο της θερμοκρασίας και θετικής μεταβολής (PTC), άνοδος της αντίστασης με την άνοδο της θερμοκρασίας.

4 Ατμοσφαιρική Πίεση

Η Ατμοσφαιρική πίεση μετρά την ποσότητα των μορίων του αέρα που βρίσκονται σε κάποιο υψόμετρο, ανάλογα με τον τρόπο και το όργανο μέτρησης σε χιλιοστά υδραργύρου (mmHg), ατμόσφαιρες (atm), ράβδοι (bar) και Pascal(Pa) καθός και υποδιαιρέσεις τους millibar(mb), hectopascal(hPa).

Μερικά από τα όργανα αυτά είναι:

Σχήμα 3: Αναλογικό Βαρόμετρο
[3]

Σχήμα 4: ψηφιακός αισθητήρας πίεσης
[11]

4.1 Μέτρηση Ατμοσφαιρικής Πίεσης

Όπως και στον αισθητήρα θερμοκρασίας έτσι και εδώ ο αισθητήρας πίεσης είναι τύπου IC (Ολοκληρομένου κυκλώματος) ο οποίος λειτουργεί με παρόμοιο τρόπο θέτοντας τις κατάλληλες τιμές στους καταχωρητές του παραμετροποιούμε το πρόγραμμα του και αλλάζουμε διάφορες παραμέτρους όπως διακριτική ικανότητα, απόκριση κ.λ.π. Ο τρόπος επικοινωνίας με τον αισθητήρα πραγματοποιείτε μέσω ενώς IC επικοινωνίας το οποίο καθορίζει και το πρωτόκολλο επικοινωνίας (I2C,SPI,UART).

4.2 Αισθητήρια Πίεσης

Όσο αφορά τα αισθητήρια πίεσης τα περισσότερα πραγματοποιούν παρόμοιες λειτουργίες, Ένας από τους μεγαλύτερους κατασκευαστές τέτοιου είδους αισθητήρων είναι η Bosch οι οποία κατασκευάζει μια σειρά αισθητήρων [7] BMP.

- Ο BMP280 είναι ένας αισθητήρας απόλυτης βαρομετρικής πίεσης ειδικά σχεδιασμένος για κινητές εφαρμογές διότι είναι εξαιρετικά συμπαγής μικρόν διαστάσεων και χαμηλής κατανάλωσης κάνοντας τον ιδανικό για χρήση με συσκευές με μπαταρίες.
- Ο BMP384 είναι σχεδιασμένος με μία μεγαλύτερη οπή πακτωμένη με ενα είδους τζέλ το οποίο τον καταστεί ανθεκτικό στο νερό και άλλα είδους χημικά κάνοντας τον ιδανικό για εφαρμογές σε αυτοματισμούς, βιομηχανικές εγκαταστάσεις, φορετές και φορητές συσκευές.
- Ο BMP390 είναι αισθητήρας χαμηλού θορύβου και χαμηλής κατανάλωσης 24bit ο οποίος είναι κατάλληλος σε εφαρμογές καταγραφής υψομετρικής διαφοράς, φορετές συσκευές, κινητά τηλέφωνα και drones.

πέρα από τα προαναφερόμενα διατίθενται και δύο νέα είδη ο BMP580 και BMP581 ο 580 ο οποίος παρέχει πολύ υψηλή ακρίβεια και ο 581 ο οποίος είναι ένας ανανεωμένος αισθητήρας που συνδυάζει όλες τις προαναφερόμενες εφαρμογές.

5 Υγρασία

Τα όργανα τα οποία μετράμε την υγρασία της ατμόσφαιρας λέγονται υγρόμετρα και το μέγεθος το οποίο καταμετρούν είναι το ποσοστό υγρασίας(νερού σε μορφή υδρατμών) ανά κυβικό αέρα δηλαδή η αναλογία αέρα-υγρασίας.

5.1 Τύποι Υγρόμετρων

Υπάρχουν πολλές τεχνικές μέτρησης της υγρασίας παρακάτω θα αναφερθούν μερικές από της σύγχρονες μεθόδους.

- Πυκνωτικά: Τα [10] πυκνωτικά υγρόμετρα όπως αναφέρει και ο τίτλος, αποτελούνται από ένα συμπαγή σε μέγεθος πυκνωτή ο οποίος είναι κατασκευασμένος από δύο ηλετρόδια με ένα λεπτό πολυμερές φιλμ, έτσι αλλάζοντας την υγρασία στον αέρα η χωρητικότητα του πυκνωτή μεταβάλετε.
- Αντίστασης: [12] Τα υγρόμετρα αντίστασης είναι κατασκευασμένα από οργανικά ημιαγώγιμα υλικά, ο ποιο συνηθισμένος τύπος ενός υγρόμετρου αντίστασης κατασκευάζεται από ένα μείγμα χλωριούχο λιθίου και άνθρακα το οποίο τοποθετείτε σε μία βάση μονωτή μεταξύ δύο ηλεκτροδίων, έτσι μεταβάλλοντας την υγρασία που βρίσκεται εκτεθειμένο το στοιχείο μεταβάλλεται και η αντίσταση του. Από 10KΩ μέχρι 10MΩ όταν η υγρασία μεταβάλλεται από 100% - 0%
- Θερμικά: Τα Θερμικά υγρόμετρα λειτουργούν με το φαινόμενο της εξάτμισης και αποτελούνται από δύο θερμόμετρα το ένα βρίσκεται εκτεθειμένο στον αέρα και το δεύτερο περικλείεται από κάποιο απορροφητικό υλικό συνήθως βαμβάκι το οποίο είναι υγρό. Έτσι με την εξάτμιση της υγρασίας από το βαμβάκι απορροφάται και ένα ποσό θερμικής ενέργειας από τον περιβάλλων αέρα και το υγρό θερμόμετρο δείχνει χαμηλότερη θερμοκρασία από το πρώτο, ο ρυθμός εξάτμισης του υγρού είναι αντιστρόφως ανάλογος με το ποσοστό υγρασίας του περιβάλλοντος δηλαδή όσο πιο ξηρός είναι ο αέρας τόσο πιο γρήγορα εξατμίζεται το υγρό παρουσιάζοντας μεγαλύτερη πτώση της θερμοκρασίας, το αντίστροφο φαινόμενο παρατηρείτε στον υγρό αέρα, έτσι με τη διαφορά αυτή των δύο θερμομέτρων μπορούμε να μετρήσουμε την σχετική υγρασία περιβάλλοντος.

6 IoT(Internet of Things)

Το Internet of Things είναι ένα σύστημα από συσκευές και αισθητήρες τα οποία είναι συνδεδεμένα στο διαδίκτυο και μεταφέρουν δεδομένα, συνδέονται μεταξύ τους και επικοινωνούν, δίνοντας μας την δυνατότητα να παρακολουθούμε και να επιδράμε σε συσκευές και αυτοματισμούς από την άνεση που μας παρέχει το διαδίκτυο μέσω οποιασδήποτε συσκευής είτε απομονομένης IoT, είτε Η/Υ όπως (κινητό τηλέφωνο, laptop, tablet, smartwatch).

Η ορολογία IoT χρησιμοποιήθηκε πρώτα από τον Kevin Ashton έναν Βρετανό πρωτοπόρο της τεχνολογίας το 1999 για να περιγράψει τις δυνατότητες διασύνδεσης των RFID ταυτοτήτων που χρησιμοποιούνται και σήμερα, για την παρακολούθηση προϊόντων μίας εταιρικής αλυσίδας δίχως την ανθρώπινη παρέμβαση.

7 Διασυνδέσεις

Οι διασυνδέσεις που υλοποιούν το δίκτυο IoT χωρίζονται στο παρακάτω διάγραμμα.

Cloud	
Internet	
Corporate Network	Privet Network
LAN | WAN	LAN | WAN
ETH / Wireless	ETH / Wireless

Αυτή η εργασία θα ασχοληθεί κυρίως στο ιδιωτικό δίκτυο μέσω ασύρματης σύνδεσης. Γενικά στον χώρο των ασύρματων διασυνδέσεων χρησιμοποιούνται πολλά πρωτόκολλα επικοινωνίας ο σταθμός σχεδιάσθηκε χρησιμοποιώντας τα πρωτόκολλα WIFI και LoRa.

7.1 LoRa

Η Ένωση LoRa (LoRa Alliance) είναι ένας μεγάλος μη κερδοσκοπικός οργανισμός ο οποίος ιδρύθηκε το 2015. Ο Σκοπός των μελών του είναι η ίδρυση και διεύρυνση το δικτύου LoRaWAN ως στάνταρ που θα χρησιμοποιείται σε εφαρμογές IoT και ανήκει στην κατηγορία LPWAN(Low Power WAN) η οποία προσφέρει ελαστικότητα λόγο της μεγάλης εμβέλειας και της χαμηλής κατανάλωσης καθώς και της ασφάλειας του δικτύου, σε όλες της εφαρμογές.

7.1.1 LoRa Διαμόρφωση

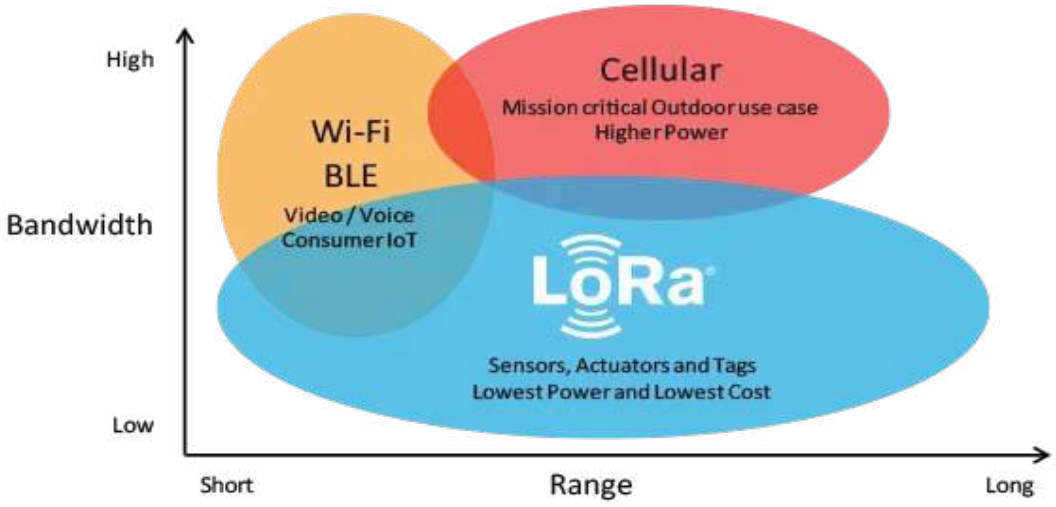
Η Διαμόρφωση LoRa είναι μια διαμόρφωση σήματος για την μεταφορά χαμηλού όγκου δεδομένων, μεγάλης απόστασης και χαμηλής κατανάλωσης, η οποία προήρθε από την τεχνική Chirp Spread Spectrum (CSS). Η Τεχνική αυτή κωδικοποιεί την πληροφορία επάνω στα ραδιοκύματα χρησιμοποιώντας παλμούς και για αυτό τον λόγο αυτοί οι παλμοί δεν μπορούν να μεταφέρουν μεγάλο όγκο δεδομένων αλλά είναι ανθεκτικοί στην παραμόρφωση και απαιτούν μικρό ποσό ενέργειας για να μεταφερθούν.

Η Αποστάσεις που μπορούν να μεταφερθούν δεδομένα είναι πολύ μεγαλύτερη σε σχέση με άλλες διαμορφώσεις όπως WiFi, Bluetooth, ZigBee.

Η LoRa διαμόρφωση λειτουργεί με δωρεάν άδεια χρήσης στο φάσμα των μεγάκυκλων (MHz) κυρίως σε ένα καταχωρημένο φάσμα το οποίο χωρίζεται ανάλογα με την γεωγραφική τοποθεσία.

EU863-870	EU868
US902-928	US915
CN779-787	CN779
EU433	EU433
AU915-928	AU915
CN470-510	CN470
AS923	AS923
KR920-923	KR920
IN865-867	IN865
RU864-870	RU864

Η Σχέση μεταξύ του εύρους φάσματος και απόστασης μεταφοράς δεδομένων είναι αντιστρόφως ανάλογη, αυτό σημαίνει πως για σταθερή ισχύ μετάδοσης του σήματος χαμηλόσυχνα σήματα μεταδίδονται σε μεγαλύτερες αποστάσεις ενώ τα υψηλόσυχνα σε μικρότερες.



Σχήμα 5: γράφημα σχέσης εύρους φάσματος - συχνότητας
[6]

7.1.2 LoRaWAN

Το [1] LoRaWAN είναι ένα πρωτόκολλο στο λογισμικό μέρος το οποίο ανήκει στο επίπεδο Media Access Contol(MAC) και λειτουργεί επάνω στη διαμόρφωση LoRa. Αυτό το πρωτόκολλο χρησιμοποιείται για να ορίσει πώς οι συσκευές θα χρησιμοποιούν το υλικό μέρος των συσκευών LoRa για να διαμορφώσουν τα μηνύματα και τα πακέτα που θα αποστέλλουν.

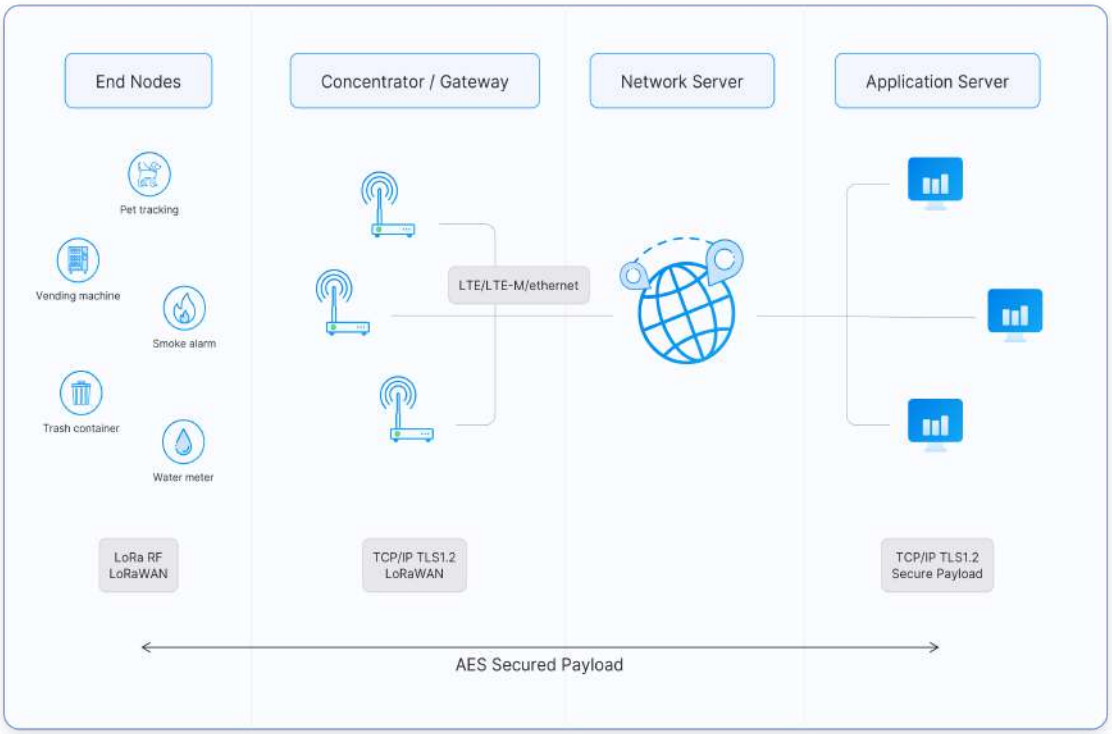
7.1.3 Αρχιτεκτονική LoRaWAN

Στο παρακάτω διάγραμμα απεικονίζεται η διασύνδεση του δικτύου το οποίο αποτελείτε από:

- Τελικές Συσκευές οι οποίες μπορεί να είναι είτε συσκευές με ενσωματωμένη λειτουργία σύνδεσης στο δίκτυο LoRaWAN και την ενσωμάτωση τους σε κάποια πλατφόρμα IoT, είτε απομονωμένοι αισθητήρες οι οποίοι μπορούν να συνδεθούν αυτούσιοι και να αποτελούν μέρος μιας ομάδας αισθητήρων.
- Πύλες (Gateways) οι οποίες λειτουργούν με παρόμοιο τρόπο όπως και τα ρούτερ τα οποία μας συνδέουν με το διαδίκτυο και συνεπώς με το IoT δίκτυο.

- Διακομιστή Δικτύου, ο Διακομιστής Δικτύου διασυνδέει το δίκτυο LoRaWAN και μεταφέρει τα δεδομένα στον διακομιστή εφαρμογών.
- Διακομιστής Εφαρμογών, από αυτό τον διακομιστή μπορούμε να δούμε άμεσα τα δεδομένα από τις τελικές συσκευές ή την διαμόρφωση τους αλλά κυρίως να μεταφέρουμε τα δεδομένα από τον διακομιστή σε κάποια εφαρμογή απεικόνισης.

Φυσικά τα δεδομένα που μεταφέρουμε μπορεί να είναι δημόσια για κοινόχρηστες εφαρμογές ή μπορεί να είναι για ιδιωτική χρήση σε αυτή την περίπτωση το δίκτυο μας δίνει την δυνατότητα να κρυπτογραφήσουμε τα δεδομένα με το προηγμένο πρότυπο κρυπτογράφησης AES κατά την δημιουργία της εφαρμογής.



Σχήμα 6: Αρχιτεκτονική του δυκτίου LoRaWAN

[13]

7.2 WiFi

Το WiFi το οποίο χρησιμοποιείτε από σχεδόν όλες της συσκευές που χρησιμοποιούμε αποτελείτε από μια ομάδα πρωτοκόλλων.

Όπως και στο LoRa έτσι και εδώ το [2] WiFi Alliance είναι ένας μη κερδοσκοπικός οργανισμός ο οποίος ιδρύθηκε το 1999, πολλές εταιρίες συγκεντρώθηκαν και ίδρυσαν αυτό τον οργανισμό έχοντας ως στόχο την λήψη στάνταρ τα οποία θα πρέπει να ακολουθούν όλες οι εταιρίες οι οποίες θέλουν να χρησιμοποιήσουν αυτή την ομάδα πρωτοκόλλων για την ομαλή διασύνδεση και χρήση τους, καθώς και την καθιέρωση και διεύρυνση του WiFi ως το κοινό μέσο διασύνδεσης τοπικών δικτύων.

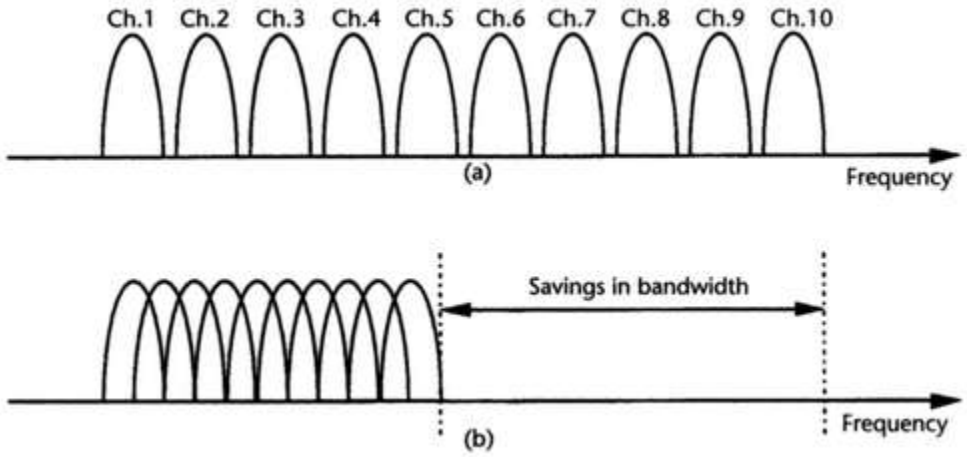
7.2.1 IEEE 802.11

Το Στάνταρ IEEE 802.11 ανήκει σε μια μεγαλύτερη ομάδα την [8] 802 η οποία καθιερώνει τα στάνταρ για όλα τα τοπικά δίκτυα, το 802.11 αφορά συγκεκριμένα τα ασύρματα σήματα τα οποία χρησιμοποιεί το WiFi και συμπεριλαμβάνει τοις συχνότητες 2.4/5/6/60 GHz. Μερικά από τα βασικά πρωτόκολλα που χρησιμοποιούμε είναι:

Generation	Protocol	Frequency	Mbit/s	Modulation	Release Date
WiFi	802.11b	2.4 GHz	1/2/5.5/11	HR-DSSS	1999
WiFi	802.11g	2.4 GHz	5/10/20	OFDM	2003
WiFi 4	802.11n	2.4/5 GHz	288.8/600	MIMO-OFDM	2009
WiFi 5	802.11ac	5 GHZ	346.8 - 3466.8	MIMO-OFDM	2013
WiFi 6/6E	802.11ax	2.4/5/6 GHz	1147 - 9608	MIMO-OFDM	2021

7.2.2 Διαμόρφωση OFDM

Η Διαμόρφωση [14] Orthogonal Frequency-Division Multiplexing (OFDM) είναι μια δημοφιλής μέθοδος ασύρματης μεταφοράς δεδομένων υψηλής ταχύτητας, το γεγονός πως είναι πολυπλεξία με ορθογώνια διαίρεση συχνότητας σημαίνει πως υπάρχει μαθηματική σχέση μεταξύ των φέρουσων συχνοτήτων του κάθε καναλιού που μας δίνει την δυνατότητα να χωρέσουμε περισσότερα κανάλια σε μικρότερο εύρος φάσματος που καθιστά την τεχνική αυτή περισσότερο αποδοτική, χωρίς να υπάρχουν παρεμβολές μεταξύ των φέρουσων.



Σχήμα 7: α) απλή πολυπλεξία β) ορθογώνια πολυπλεξία
[15]

8 Σχεδιασμός κυκλώματος

Το κύκλωμα του μετεωρολογικού σταθμού έχει τρία μεγέθη συλλογής δεδομένων.

- - Θερμοκρασία
- - Σχετική Υγρασία
- - Πίεση

Καθώς και μία οθόνη LCD 16 κελιών και 2 σειρών, WiFi μικροελεγκτής και τέλος ο κεντρικός μικροελεγκτής [5] Arduino UNO που διαχειρίζεται όλα τα modules.

8.1 Αισθητήρες Θεμοκρασίας Πίεσης και LCD

Οι αισθητήρες Θερμοκρασίας και πίεσης καθώς και η οθόνη είναι συνδεμένοι με το Arduino με το πρωτόκολλο σειριακής επικοινωνίας I2C το οποίο χρησιμοποιεί δύο αγωγούς επικοινωνίας (SDA, SCL) δεδομένων και ρολογιού και επικοινωνεί με το κάθε module στέλνοντας πρώτα την διεύθυνση με την οποία θέλει να ανταλλάξει δεδομένα με την μορφή δεκαεξαδικού αριθμού κάθε συσκευή που είναι συνδεδεμένη έχει ξεχωριστή διεύθυνση για να μπορούμε να επιλέξουμε μόνο αυτή χωρίς να υπάρχει μπέρδεμα κατά τη διάρκεια την επικοινωνίας

8.2 Αισθητήρας Σχετικής Υγρασίας

Ο Αισθητήρας σχετικής υγρασίας δεν επικοινωνεί με το πρωτόκολλο I2C αλλά με το One Wire, Two Way το οποίο χρησιμοποιεί μόνο έναν αγωγό στον οποίο πρώτα πραγματοποιείτε συγχρονισμός μεταξύ τους δηλαδή πρώτα στέλνει ο μικροελεγκτής ένα σήμα εκκίνησης και ο αισθητήρας μεταβαίνει από κατάσταση χαμηλής κατανάλωσης σε κατάσταση κανονικής λειτουργίας και απαντά πίσω ένα μήνημα 40-Bit που περιέχει την μέτρηση της υγρασίας.

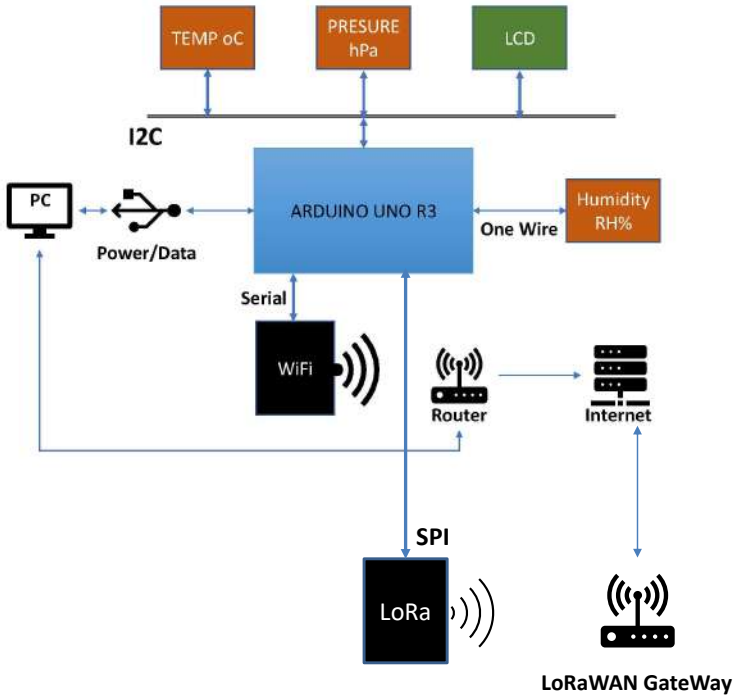
8.3 WiFi Module

Η Πλακέτα του WiFi επικοινωνεί με τον μικροελεγκτή μέσω σειριακής επικοινωνίας (Serial) αφού επιλεχθεί η ταχύτητα συγχρονισμού από τις δύο συσκευές ξεκινούν την μεταφορά δεδομένων και οι δύο συσκευές έχουν έναν καταχωρητή ο οποίος χρησιμοποιείτε για την προσωρινή αποθήκευση κάθε λέξης που μεταφέρεται και με την κατάλληλη δομή επανάληψης στο προγραμματισμό του μικροελεγκτή μεταφέρουμε τα δεδομένα από τον καταχωρητή στη μνήμη, όταν σταλθεί το κατάλληλο σύμβολο στο τέλος της λέξης που μεταδόθηκε αναγνωρίζουμε ότι ολοκληρώθηκε η μετάδοση της λέξης, η σειριακή επικοινωνία λειτουργεί μόνο με την διάταξη (Master - Slave).

8.4 LoRa Module

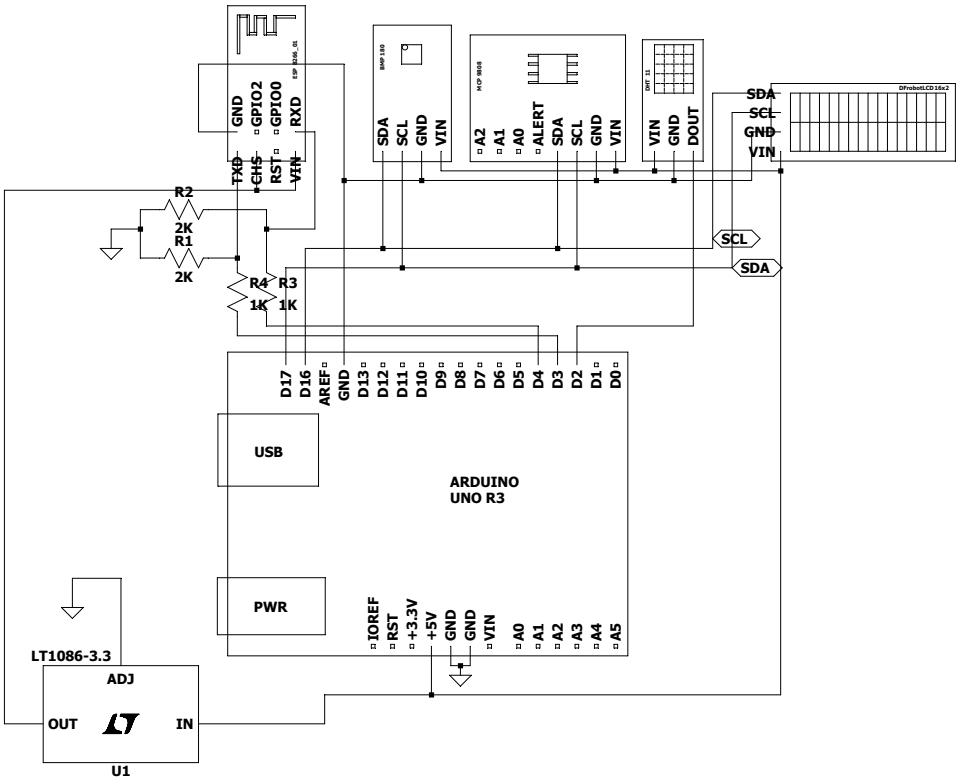
Η Πλακέτα LoRa χρησιμοποιεί το πρωτόκολλο SPI(Serial Peripheral Interface) για την επικοινωνία της με τον μικροελεγκτή. Το πρωτόκολλο αυτό χρησιμοποιεί τέσσερις αγωγούς για την σύνδεση οι οποίοι είναι MISO, MOSI, CSS, CLK (Master In Slave Out, Master Out Slave In, Chip Select, Clock), η επικοινωνία αυτή μπορεί να διαχειριστή περισσότερες συσκευές από τα προηγούμενα πρωτόκολλα διότι ο μόνος αγωγός που χρειάζεται για να επιλέξουμε συσκευή είναι ο Chip Select όλοι οι άλλοι παραμένουν συνδεμένοι στο ίδιο BUS.

8.5 FBD Μετεωρολογικού σταθμού



Σχήμα 8: Function Block Diagram

8.6 Κυκλωματικό Διάγραμμα



Σχήμα 9: κύκλωμα σταθμού

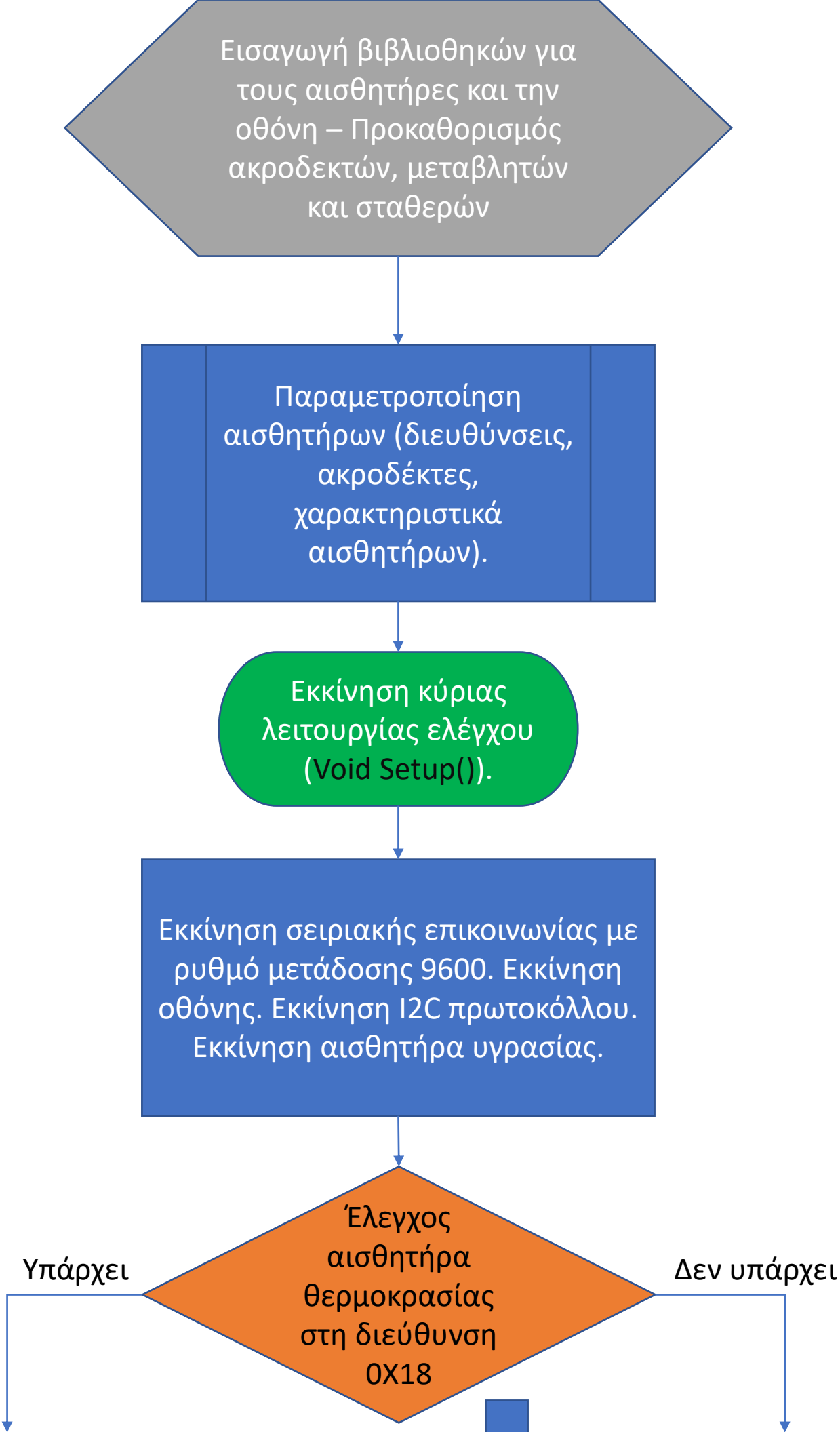
9 Προγραμματισμός Σταθμού

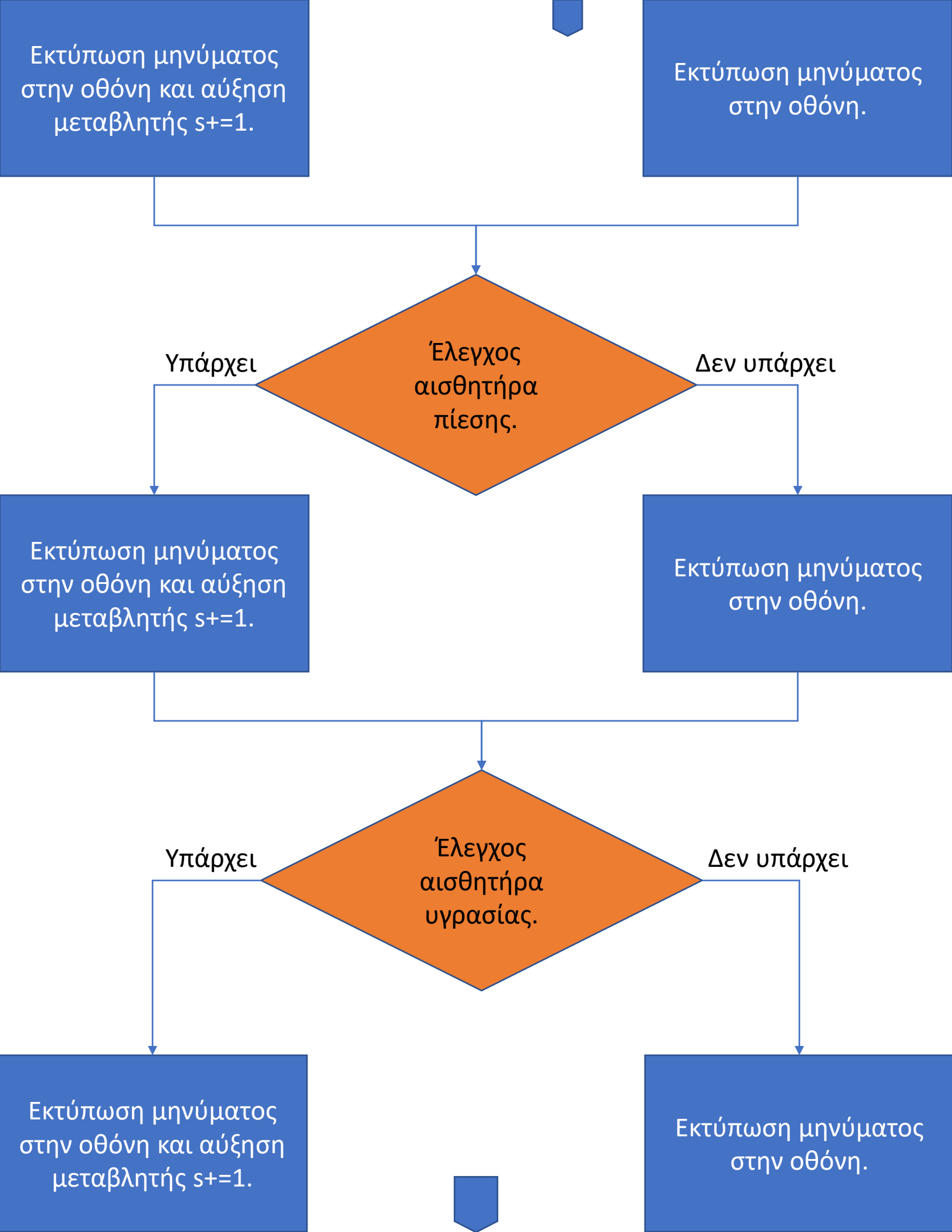
Ο προγραμματισμός θα αναληθή σε δύο μέρη πρώτα με Flow Chart και έπειτα σε κείμενο, και για τον μικροελεγκτή καθώς και για το WiFi Module.

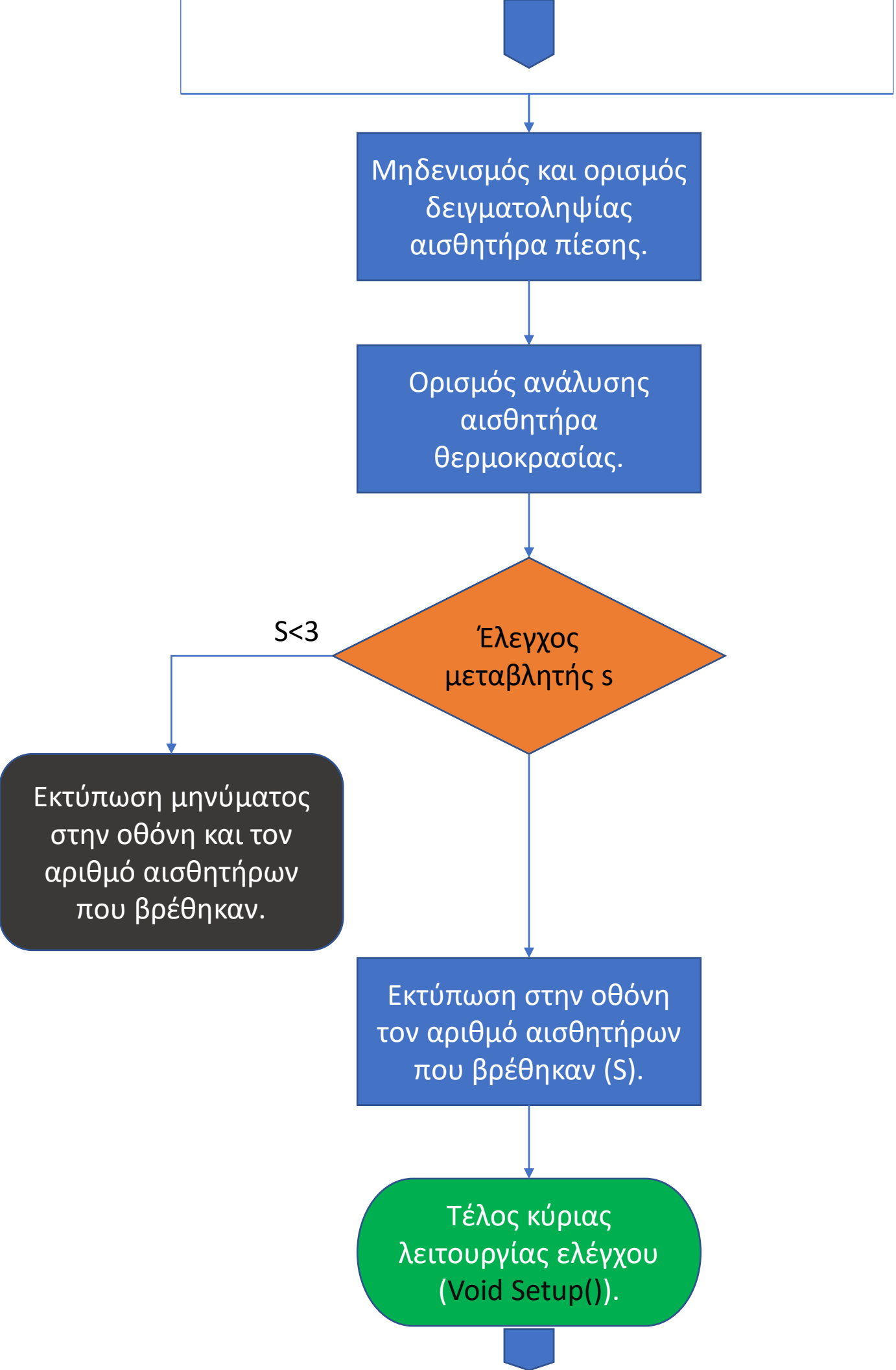
9.1 Προγραμματισμός Arduino

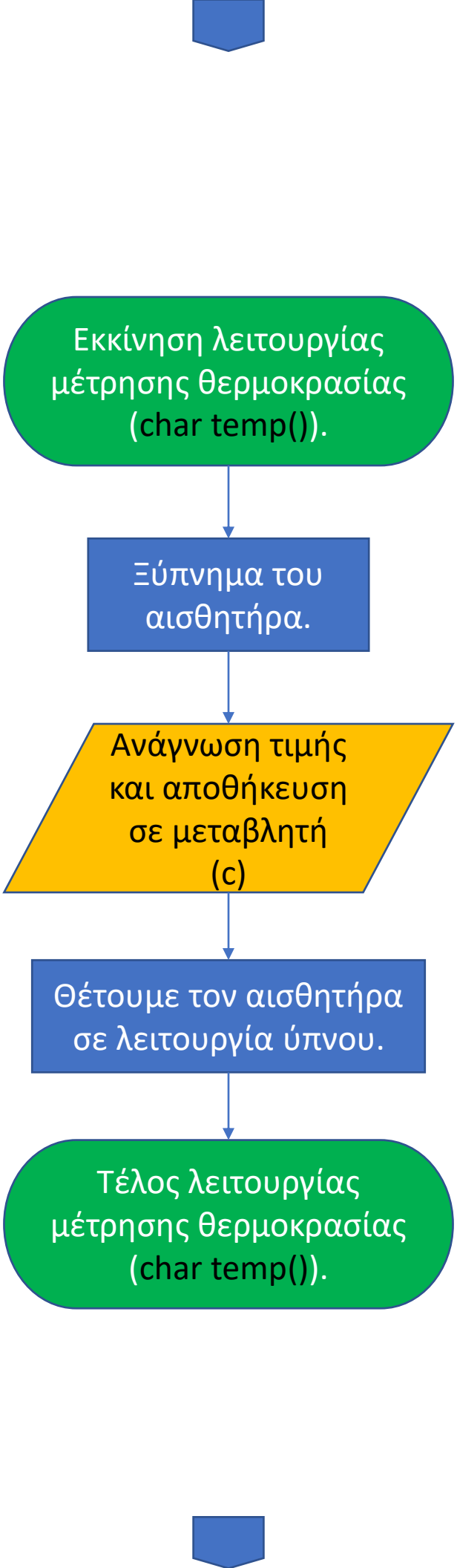
Για τον προγραμματισμό του μικροελεγκτή οι λειτουργίες που απαιτούνται να πραγματοποιήσει είναι η επικοινωνία με τους αισθητήρες, η συλλογή των δεδομένων και η μετάδωση τους στο WiFi Module, για αυτό τον λόγο στη σειριακή επικοινωνία μεταξύ τους ο μικροελεγκτής θα παριστά τον ρόλο του (Slave).

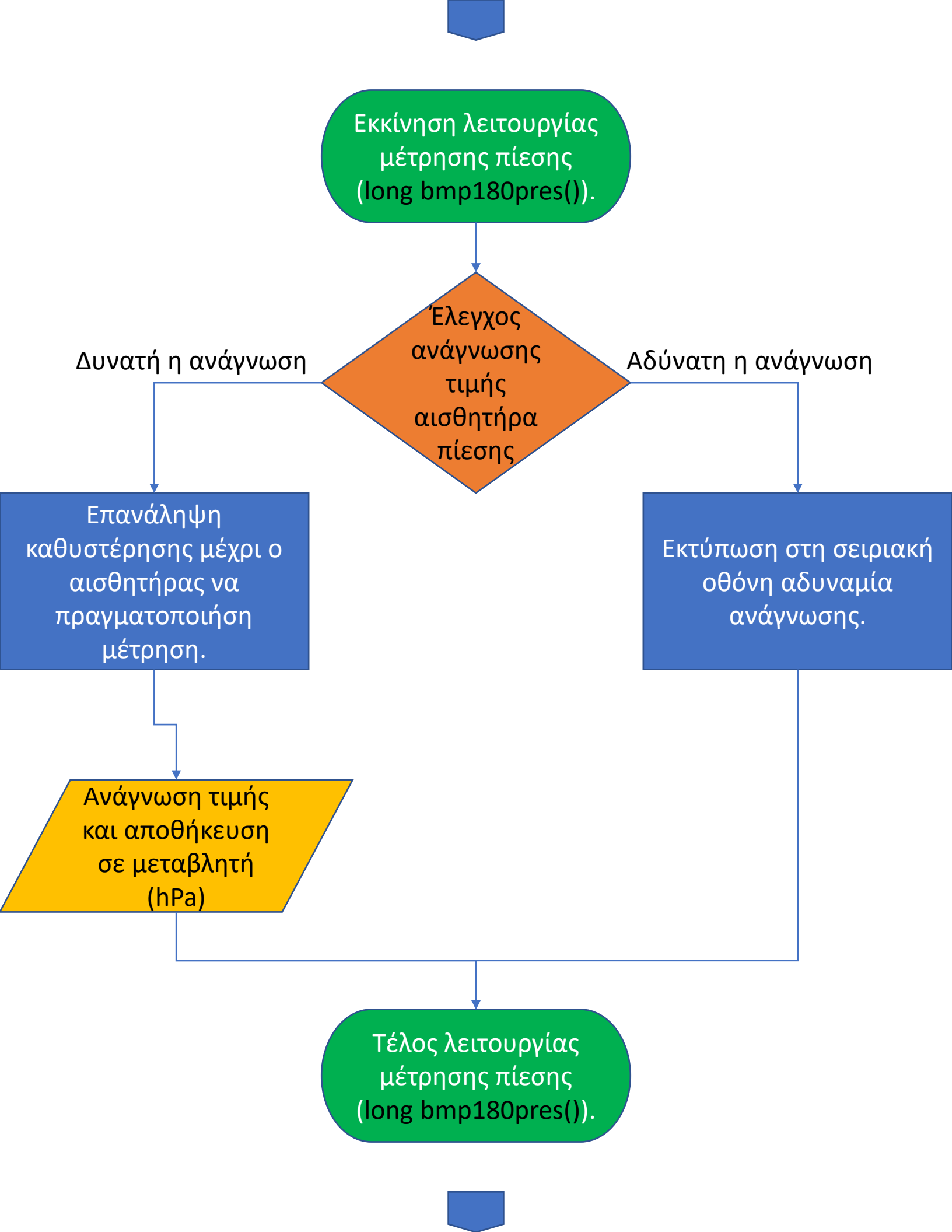
9.1.1 Κώδικας Flow Chart

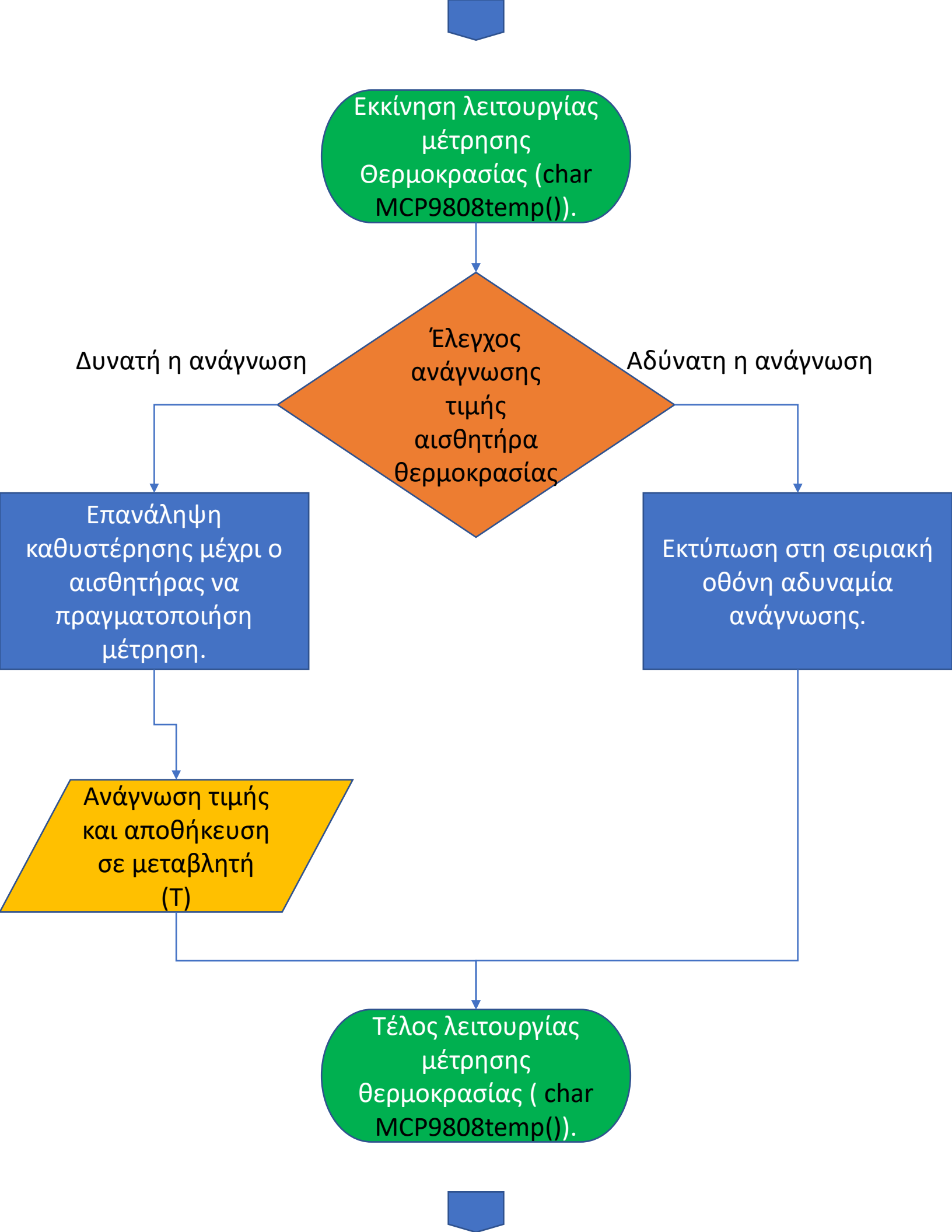

Εισαγωγή βιβλιοθηκών για τους αισθητήρες και την οθόνη – Προκαθορισμός ακροδεκτών, μεταβλητών και σταθερών
Παραμετροποίηση αισθητήρων (διευθύνσεις, ακροδέκτες, χαρακτηριστικά αισθητήρων).
Εκκίνηση κύριας λειτουργίας ελέγχου (Void Setup()).
Εκκίνηση σειριακής επικοινωνίας με ρυθμό μετάδοσης 9600. Εκκίνηση οθόνης. Εκκίνηση I2C πρωτοκόλλου. Εκκίνηση αισθητήρα υγρασίας.
Έλεγχος αισθητήρα θερμοκρασίας στη διεύθυνση 0X18
Υπάρχει
Δεν υπάρχει

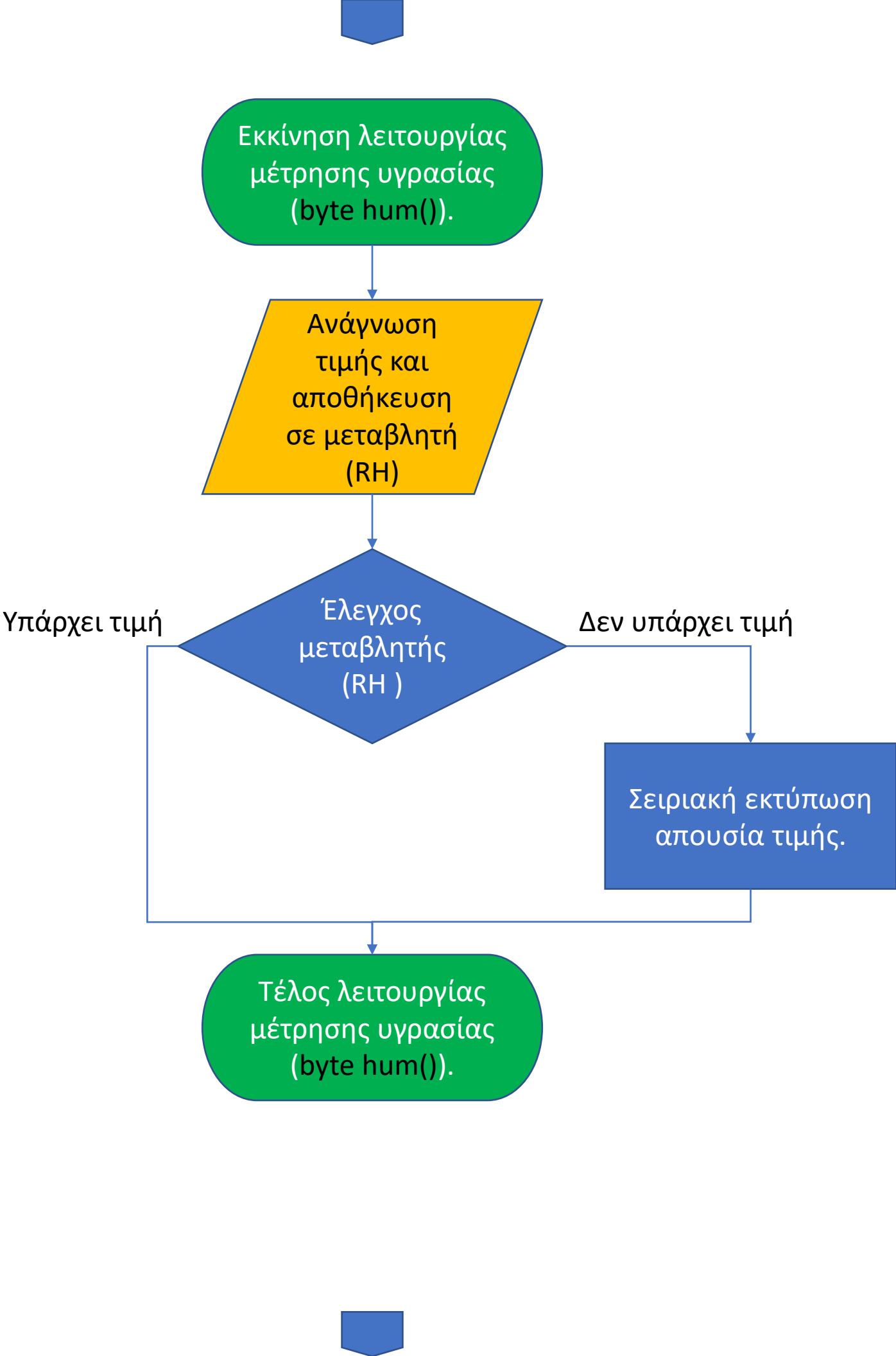

Εκτύπωση μηνύματος στην οθόνη και αύξηση μεταβλητής s+=1.
Εκτύπωση μηνύματος στην οθόνη.
Έλεγχος αισθητήρα πίεσης.
Υπάρχει
Δεν υπάρχει
Εκτύπωση μηνύματος στην οθόνη και αύξηση μεταβλητής s+=1.
Εκτύπωση μηνύματος στην οθόνη.
Έλεγχος αισθητήρα υγρασίας.
Υπάρχει
Δεν υπάρχει
Εκτύπωση μηνύματος στην οθόνη και αύξηση μεταβλητής s+=1.
Εκτύπωση μηνύματος στην οθόνη.

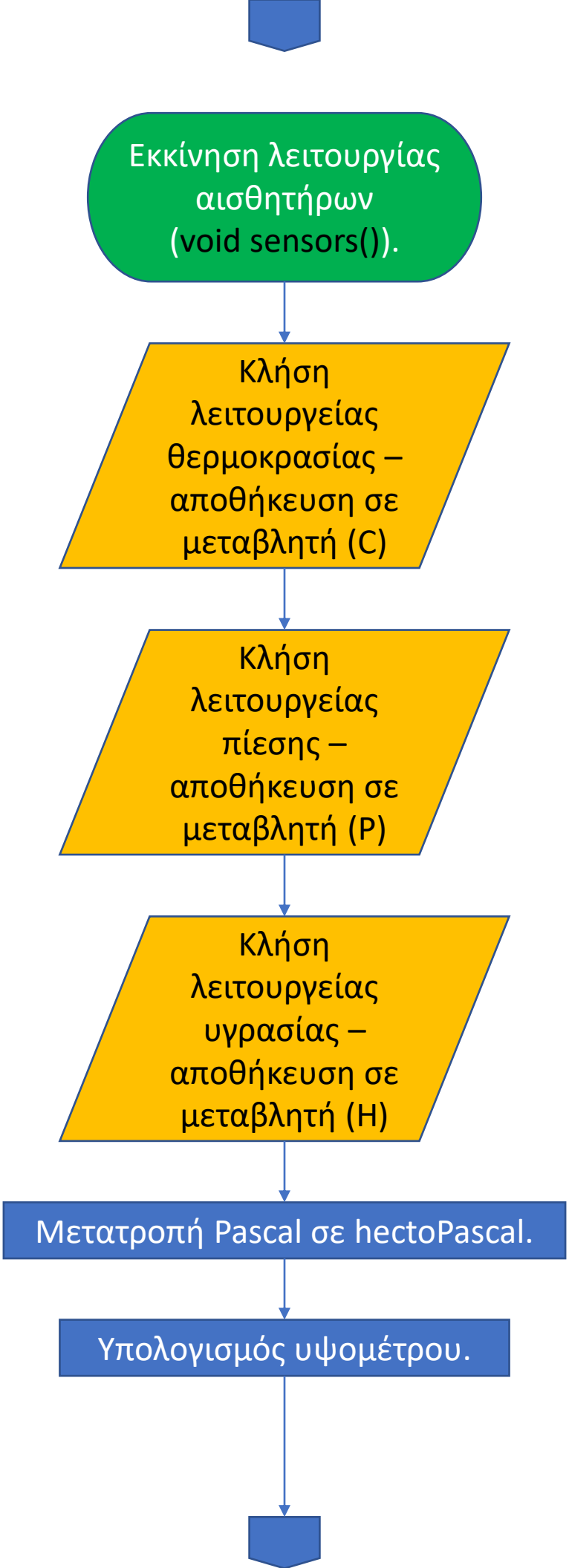

Μηδενισμός και ορισμός δειγματοληψίας αισθητήρα πίεσης.
Ορισμός ανάλυσης αισθητήρα θερμοκρασίας.
Έλεγχος μεταβλητής s
S<3
Εκτύπωση μηνύματος στην οθόνη και τον αριθμό αισθητήρων που βρέθηκαν.
Εκτύπωση στην οθόνη τον αριθμό αισθητήρων που βρέθηκαν (S).
Τέλος κύριας λειτουργίας ελέγχου (Void Setup()).

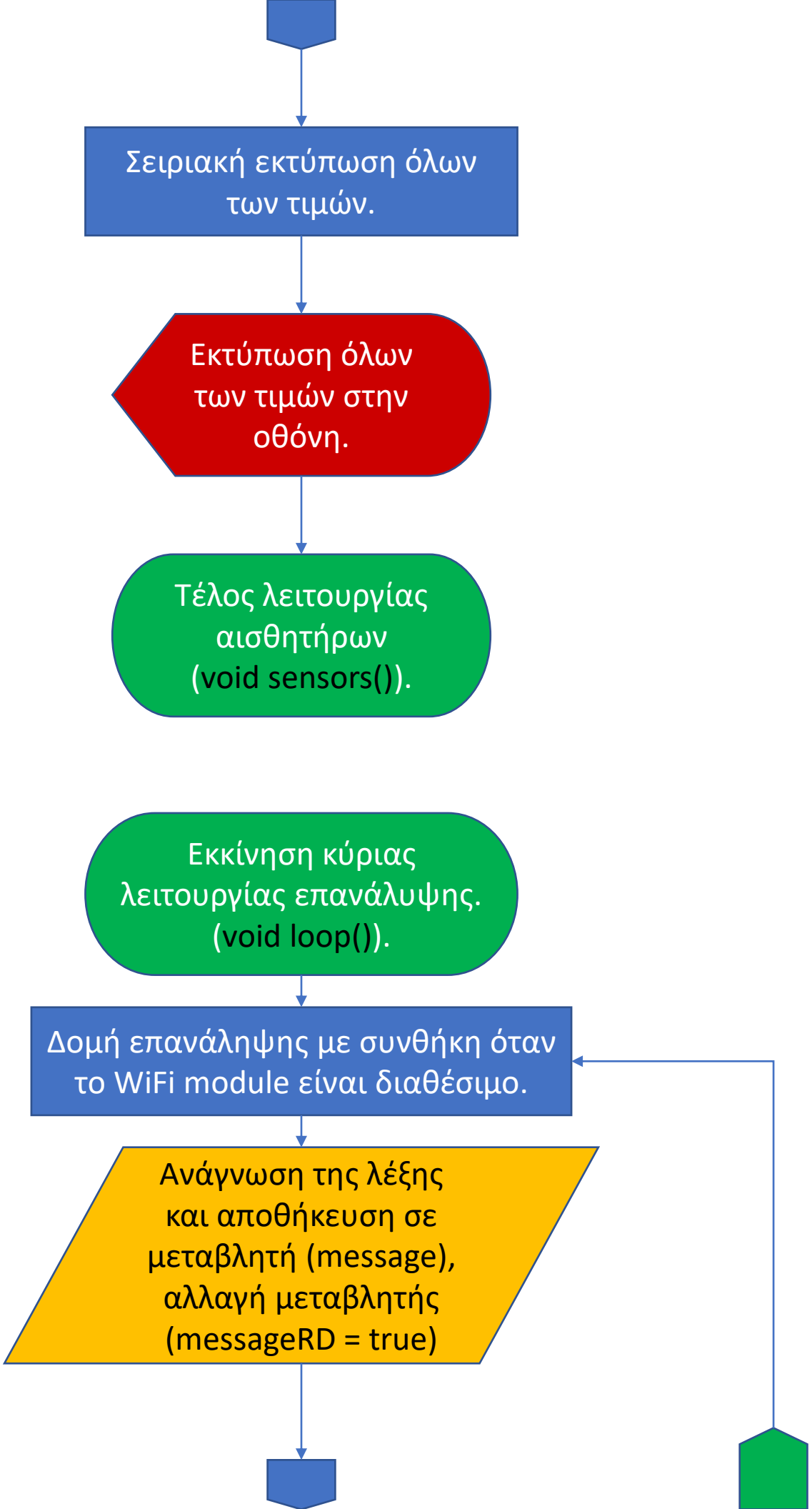

Εκκίνηση λειτουργίας μέτρησης θερμοκρασίας (char temp()).
Ξύπνημα του αισθητήρα.
Ανάγνωση τιμής και αποθήκευση σε μεταβλητή (c)
Θέτουμε τον αισθητήρα σε λειτουργία ύπνου.
Τέλος λειτουργίας μέτρησης θερμοκρασίας (char temp()).

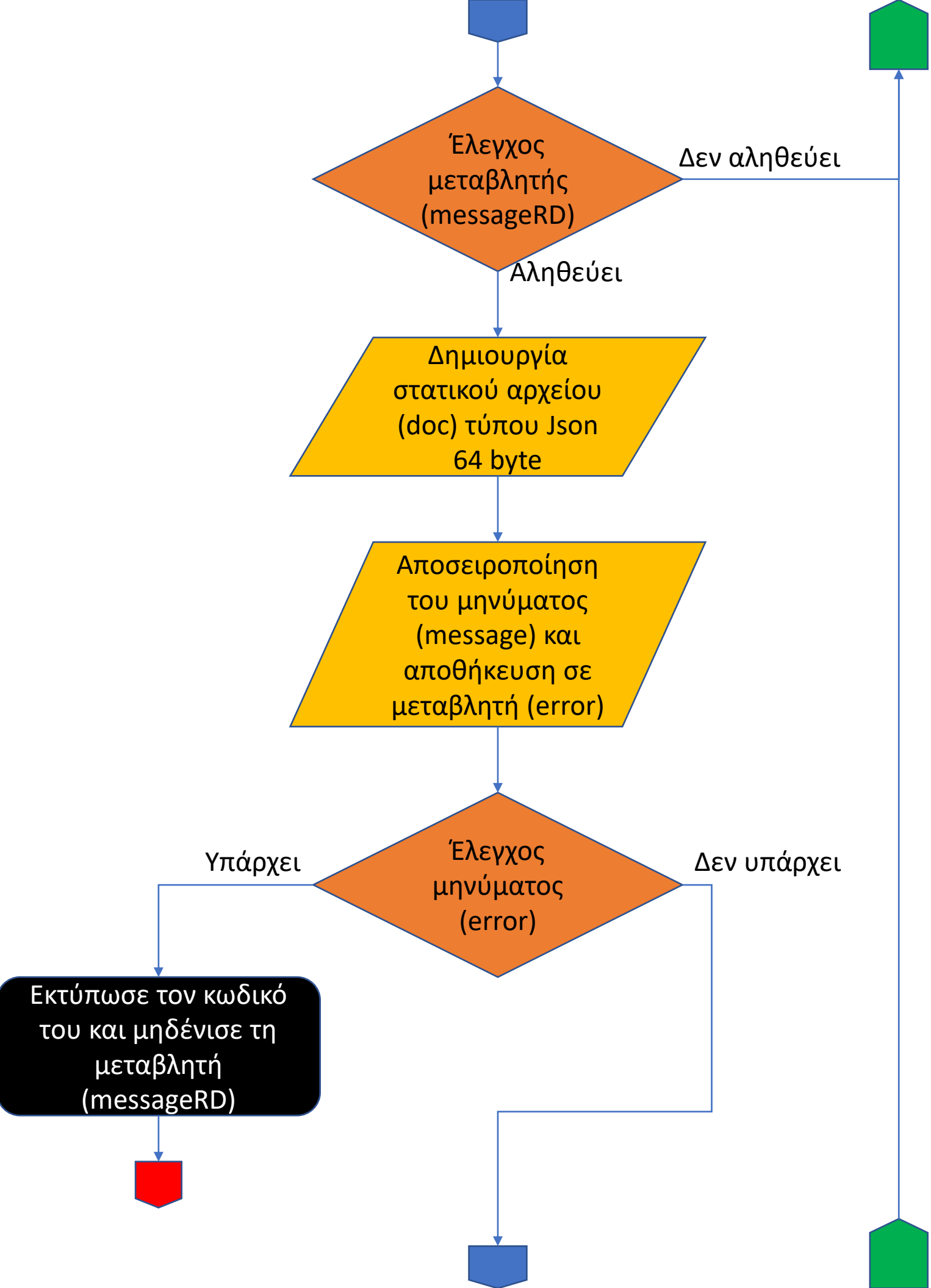

Εκκίνηση λειτουργίας μέτρησης πίεσης (long bmp180pres()).
Έλεγχος ανάγνωσης τιμής αισθητήρα πίεσης
Δυνατή η ανάγνωση
Αδύνατη η ανάγνωση
Επανάληψη καθυστέρησης μέχρι ο αισθητήρας να πραγματοποιήση μέτρηση.
Εκτύπωση στη σειριακή οθόνη αδυναμία ανάγνωσης.
Ανάγνωση τιμής και αποθήκευση σε μεταβλητή (hPa)
Τέλος λειτουργίας μέτρησης πίεσης (long bmp180pres()).

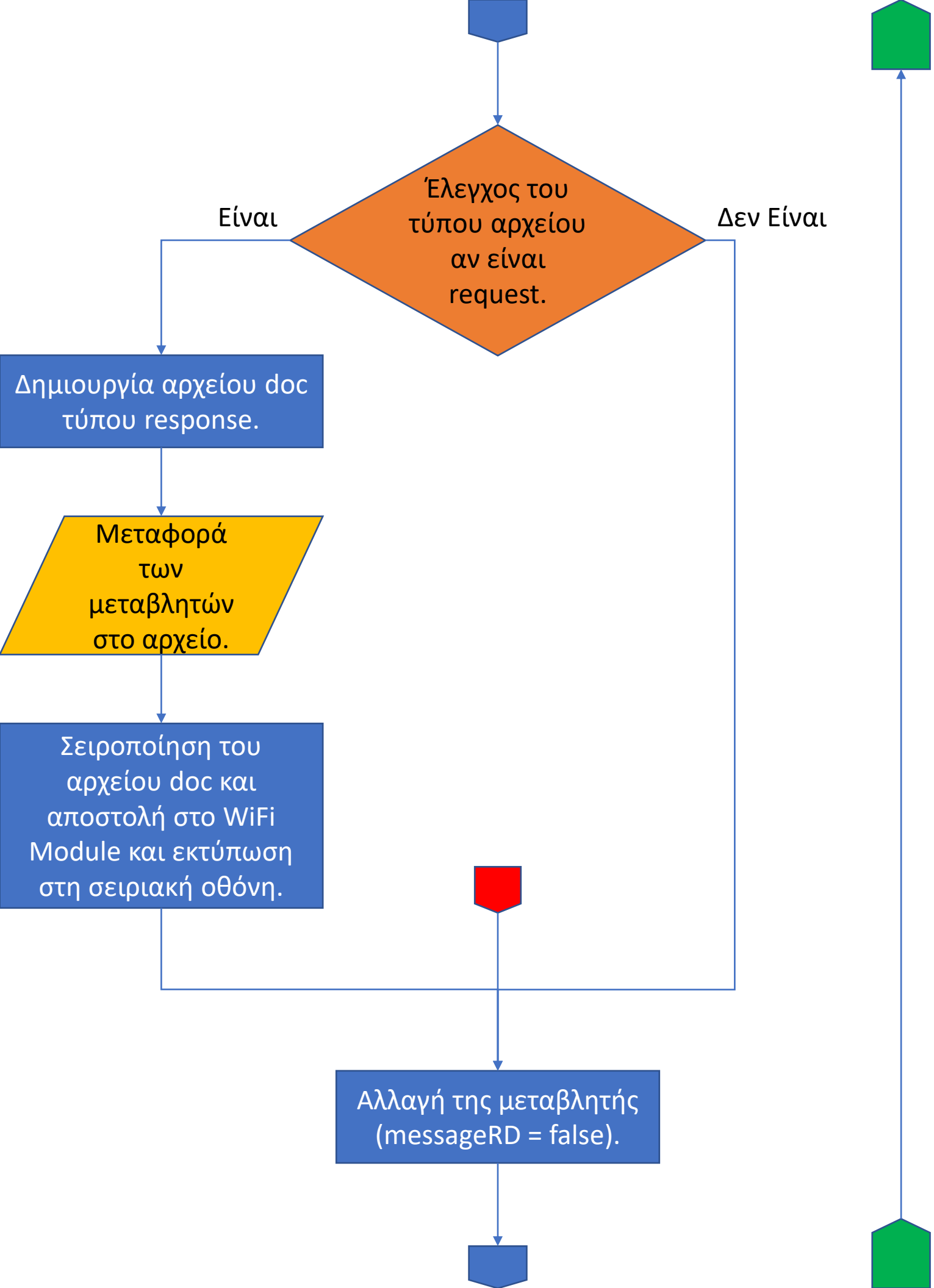

Εκκίνηση λειτουργίας μέτρησης Θερμοκρασίας (char MCP9808temp()).
Έλεγχος ανάγνωσης τιμής αισθητήρα θερμοκρασίας
Δυνατή η ανάγνωση
Αδύνατη η ανάγνωση
Επανάληψη καθυστέρησης μέχρι ο αισθητήρας να πραγματοποιήση μέτρηση.
Εκτύπωση στη σειριακή οθόνη αδυναμία ανάγνωσης.
Ανάγνωση τιμής και αποθήκευση σε μεταβλητή (T)
Τέλος λειτουργίας μέτρησης θερμοκρασίας (char MCP9808temp()).

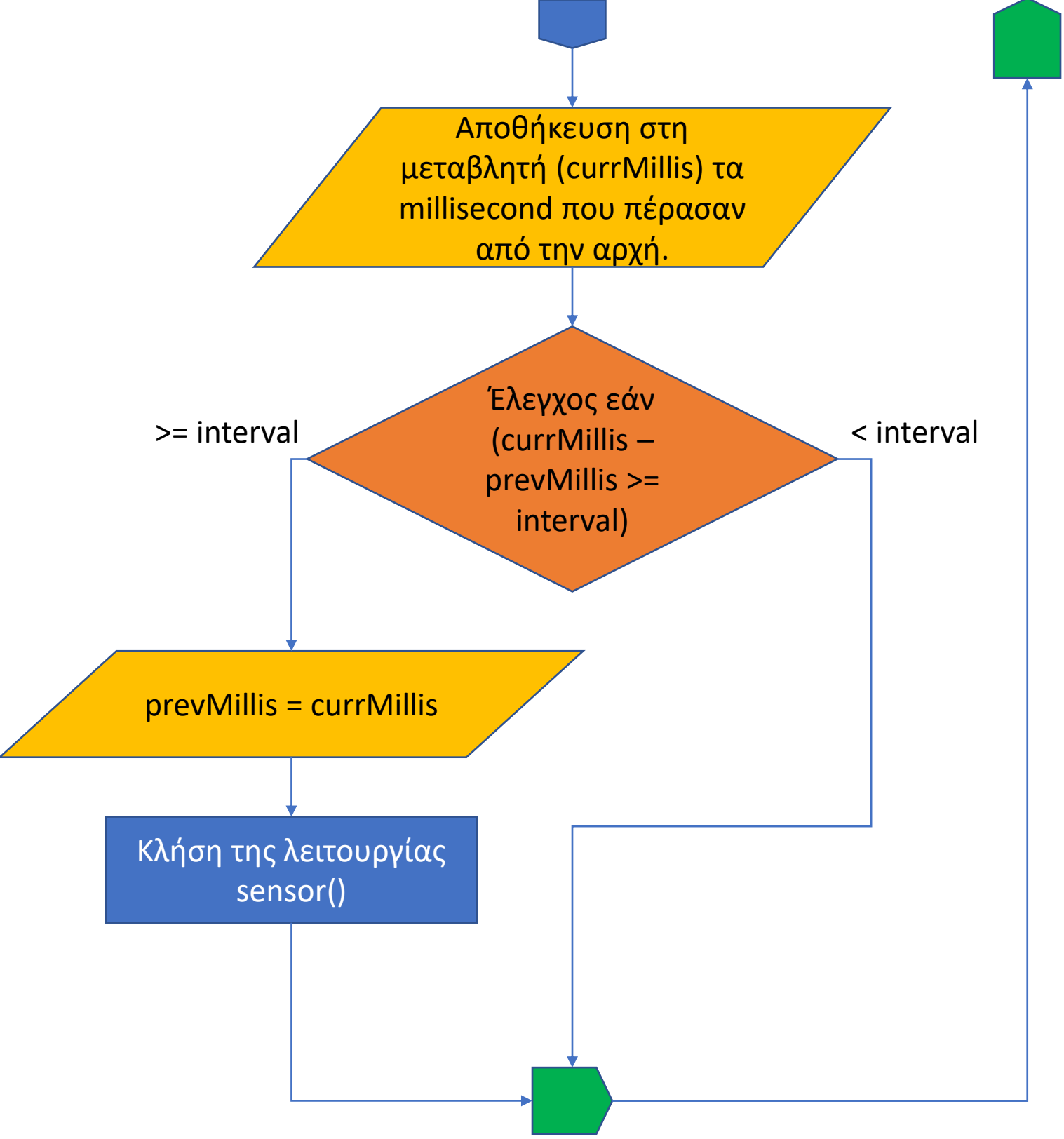

Εκκίνηση λειτουργίας μέτρησης υγρασίας (byte hum()).
Ανάγνωση τιμής και αποθήκευση σε μεταβλητή (RH)
Έλεγχος μεταβλητής (RH)
Υπάρχει τιμή
Δεν υπάρχει τιμή
Σειριακή εκτύπωση απουσία τιμής.
Τέλος λειτουργίας μέτρησης υγρασίας (byte hum()).


Εκκίνηση λειτουργίας αισθητήρων (void sensors()).
Κλήση λειτουργείας θερμοκρασίας – αποθήκευση σε μεταβλητή (C)
Κλήση λειτουργείας πίεσης – αποθήκευση σε μεταβλητή (P)
Κλήση λειτουργείας υγρασίας – αποθήκευση σε μεταβλητή (H)
Μετατροπή Pascal σε hectoPascal.
Υπολογισμός υψομέτρου.


Σειριακή εκτύπωση όλων των τιμών.
Εκτύπωση όλων των τιμών στην οθόνη.
Τέλος λειτουργίας αισθητήρων (void sensors()).
Εκκίνηση κύριας λειτουργίας επανάλυψης. (void loop()).
Δομή επανάληψης με συνθήκη όταν το WiFi module είναι διαθέσιμο.
Ανάγνωση της λέξης και αποθήκευση σε μεταβλητή (message), αλλαγή μεταβλητής (messageRD = true)


Έλεγχος μεταβλητής (messageRD)
Δεν αληθεύει
Αληθεύει
Δημιουργία στατικού αρχείου (doc) τύπου Json 64 byte
Αποσειροποίηση του μηνύματος (message) και αποθήκευση σε μεταβλητή (error)
Έλεγχος μηνύματος (error)
Υπάρχει
Δεν υπάρχει
Εκτύπωσε τον κωδικό του και μηδένισε τη μεταβλητή (messageRD)


Έλεγχος του τύπου αρχείου αν είναι request.
Είναι
Δεν Είναι
Δημιουργία αρχείου doc τύπου response.
Μεταφορά των μεταβλητών στο αρχείο.
Σειροποίηση του αρχείου doc και αποστολή στο WiFi Module και εκτύπωση στη σειριακή οθόνη.
Αλλαγή της μεταβλητής (messageRD = false).


Αποθήκευση στη μεταβλητή (currMillis) τα millisecond που πέρασαν από την αρχή.
Έλεγχος εάν (currMillis – prevMillis >= interval)
>= interval
< interval
prevMillis = currMillis
Κλήση της λειτουργίας sensor()

9.1.2 Κώδικας Text

```
#include <Wire.h>
#include <BMP180I2C.h>
#include <SoftwareSerial.h>
#include <ArduinoJson.h>
#include "DHT.h"
#include "Adafruit_MCP9808.h"
#include "DFRobot_RGBLCD1602.h"
#include <Math.h>

#define RXpin 4
#define TXpin 3
#define dht_pin 2

String message = "";//variable for the request string from esp
bool messageRD = false;//boolean variable when the there is a mesage

unsigned long prevMills = 0;//previous run time
unsigned long currMills = 0;//current run time
const long interval = 600000;//10min constant delay

char C = 0;//temperature value type
long P = 0;//pressure value type
byte H = 0;//humidity value type
```

```
long P0 = 1013.25; //Altitude at sea level in hPa

long Pw = 0.190; //Constant


byte s = 0;  //sensors variable


//BMP180 Preasure sensor I2C adress 0x77//

//MCP9808 Temp sensor I2C adress 0x18//


//specify sensor objects and parameters//

DFRobot_RGBLCD1602 lcd(16, 2); //LCD with 16 characters and 2 lines

DHT dht(dht_pin, DHT11);  //Specify the humidity sensor data pin at
pin 2

Adafruit_MCP9808 tempsensor = Adafruit_MCP9808(); // Create the
MCP9808 temperature sensor object

BMP180I2C bmp180(0x77);  //Create the BMP180 Presure sensor object

SoftwareSerial esp(RXpin, TXpin); //define the software serial port at
pre defined pins

//--oo--//




//initial setup code after turning ON//

void setup() {


  Serial.begin(9600); //Begin serial monitor (mainly for debuging)

  esp.begin(9600); //Begin esp software serial port at 9600baud
```

```
  //LCD Init//
  lcd.init(); //initialize the lcd
  lcd.display();  //turn on the display
  lcd.setColor(GREEN);
  delay(500);

  lcd.print("Booting...");  //Print startup mesage
  delay(1000);
  lcd.clear();  //Clear the display
  //--oo--//

  while (!Serial);
  Wire.begin(); //Begin I2C comunication Interface
  dht.begin();  //initialyze the humidity sensor

  //Check for sensor MCP9808//
  if (tempsensor.begin(0x18) == false)  //Test condition for temp
sensor detection
  {

    Serial.println("Couldn't find MCP9808! Check your MCP9808
Interface and I2C Address.");
    lcd.print("Check MCP9808");
    delay(1000);
```

```
  }
  else
  { //Found condition

    Serial.println("Found MCP9808!");
    lcd.home();
    lcd.print("Found MCP9808!");
    delay(1000);
    lcd.clear();
    s += 1;
  }
  //-------------------------------------------------------------------
----------------------//

  //Check for sensor BMP180//
  if (bmp180.begin() == false)  //Test condition for pressure sensor
detection
  {

    Serial.println("Couldn't find BMP180!. Check your BMP180 Interface
and I2C Address.");
    lcd.setCursor(0, 2);
    lcd.print("Check BMP180");
    delay(1000);
```

```
  }
  else  //Found condition
  {

    Serial.println("Found BMP180!");
    lcd.home();
    lcd.print("Found BMP180!");
    delay(1000);
    lcd.clear();
    s += 1;
  }
  //-------------------------------------------------------------------
--------------------//

  //Check for sensor DHT11//
  if (isnan(dht.readHumidity()))
  {

    Serial.println("Couldn't find DHT11!. Check your DHT11
Interface.");
    lcd.clear();
    lcd.home();
    lcd.print("Check DHT11");
    delay(1000);
  }
```

```
  else  //Found condition
  {

    Serial.println("Found DHT11!");
    lcd.home();
    lcd.print("Found DHT11!");
    s += 1;
  }
  //-------------------------------------------------------------------
--------------------//



  bmp180.resetToDefaults(); //zero the sensor registers


  bmp180.setSamplingMode(BMP180I2C::MODE_STD); //set resolution mode
for pressure measurements
  // Mode Parameter Samples SampleTime
  // ULP      0        1      4.5ms
  // STD      1        2      7.5ms
  // HR       2        4      13.5ms
  // UHR      3        8      25.5ms



  tempsensor.setResolution(1);  //set the temp sensor resolution
```

```
// Mode Resolution SampleTime
//  0    0.5°C       30 ms
//  1    0.25°C      65 ms
//  2    0.125°C     130 ms
//  3    0.0625°C    250 ms

delay(1000);

if (s < 3)
{

  lcd.clear();
  lcd.print("Check Sensors!");
  lcd.setCursor(0, 1);
  lcd.print("Found");
  lcd.setCursor(6, 1);
  lcd.print(s);
  lcd.setCursor(7, 1);
  lcd.print("/3 reset");

  do {

    delay(500);
  } while (s < 3);
```

```
  }
  Serial.print("Found :");
  Serial.println(s);

  sensors();//run sensors function 1 time
}

//Read TEMP function from mcp9808 sensor//
char temp() {

  tempsensor.wake();  //wake the sensor
  char c = tempsensor.readTempC(); //call for a measurement
  tempsensor.shutdown_wake(1);  //shutdown the sensor

  return c;
}

//Read PRESSURE function from bmp180 sensor//
long bmp180pres()
{

  long hPa = 0; //set local variable with the correct data type.
```

```
  //start a preasure measurement//

  if (!bmp180.measurePressure())

  {

    Serial.println("could not start perssure measurement, Couldn't
find BMP180!");

  }



  //wait for the measurement to finish. proceed as soon as hasValue()
returned true.

  do

  {

    delay(100);

  } while (!bmp180.hasValue());



  hPa = bmp180.readPressure(); //Move pressure to a variable

  //----------------------------//

  return hPa; //return preasure variable.

}



//Read TEMPERATURE function from bmp180 sensor//

char bmp180temp()

{



  char T = 0;
```

```
  //start a temperature measurement//
  if (!bmp180.measureTemperature())
  {
    Serial.println("could not start temperature measurement, Couldn't find BMP180!.");
  }

  //wait for the measurement to finish. proceed as soon as hasValue() returned true.
  do
  {
    delay(100);
  } while (!bmp180.hasValue());

  T = bmp180.readTemperature();  //Move temp reading to variable
  //-----------------------------//
  return T;  //return temperature variable
}

//Read humidity sensor function//
byte hum() {
  byte RH = dht.readHumidity();
  if  (isnan(RH))
  {
    Serial.print("No data DHT11");
```

```
    return 0;
  }
  return RH;
}

void sensors()
{
  C = temp(); //MCP9808 temp sensor call in to variable
  P = bmp180pres();  //BMP180 pressure sensor call in to variable
  H = hum();  //DHT11 humidity sensor call in to variable

  P = P / 100; //Convert Pascal in hectoPascal

  //conversion equation from pressure to altitude//
  long alt = (pow((P0/P), Pw) - 1)*(C+273.15) / 0.0065;

  //serial print for debuging//
  Serial.print(C, DEC); //Temp
  Serial.print("oC\n");

  Serial.print(P, DEC); //Pressure
  Serial.print(" :hPa \n");

  Serial.print(alt);  //Altitude
```

```
Serial.print(" :alt \n");


Serial.print(H, DEC); //Humidity
Serial.print(" :%RH \n");
//--oo--//


lcd.clear();
lcd.home();


//Print the temperature
lcd.print(C, DEC);
lcd.setCursor(3, 0);
lcd.print("C");


//Print the Humidity
lcd.setCursor(5, 0);
lcd.print(H, DEC);
lcd.print("%Rh");


//Print the altitude
lcd.setCursor(11, 0);
lcd.print(alt);
lcd.setCursor(15, 0);
lcd.print("m");
```

```
  //Print the Pressure
  lcd.setCursor(0, 1);
  lcd.print(P, DEC);
  lcd.setCursor(4, 1);
  lcd.print("hPa");
}

void loop()
{
//Constantly check if there is a message from ESP8266//
  while (esp.available())
  {
    message = esp.readString();
    messageRD = true;
  }

//If there is a message execute the Json code//
  if (messageRD)
  {
    StaticJsonDocument<64> doc;

    //Deserialize the message in an 'error' doc variable type//
    DeserializationError error = deserializeJson(doc, message);
```

```
    if (error)//if there is an error print the error code
    {
      Serial.print(F("ERROR: "));
      Serial.println(error.c_str());
      messageRD = false;
      return;
    }
    if (doc["type"] == "request")//if there is a request type message
    {
      //generate a response message with the variables we want to send
in json format//
      //and serialize the message//
      doc["type"] = "response";

      doc["tmp"] = C;
      doc["hum"] = H;
      doc["pres"] = P;
      serializeJson(doc, esp);
      serializeJson(doc, Serial);
      Serial.println();
    }
    messageRD = false;
  }
```

```
  currMills = millis();

//Time evaluation for 10 min to call sensors function without
interupting the main loop//

  if (currMills - prevMills >= interval)

  {

    prevMills = currMills;



    sensors();

  }

}
```

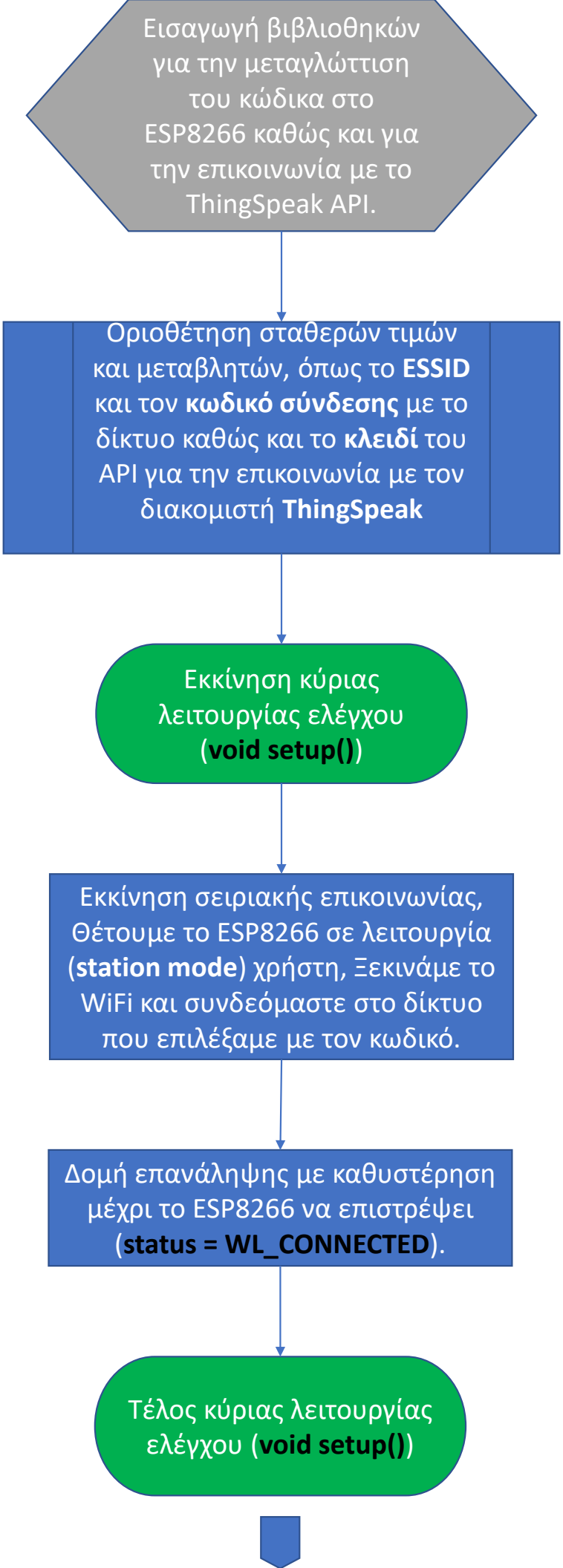
9.2 Προγραμματισμός ESP8266

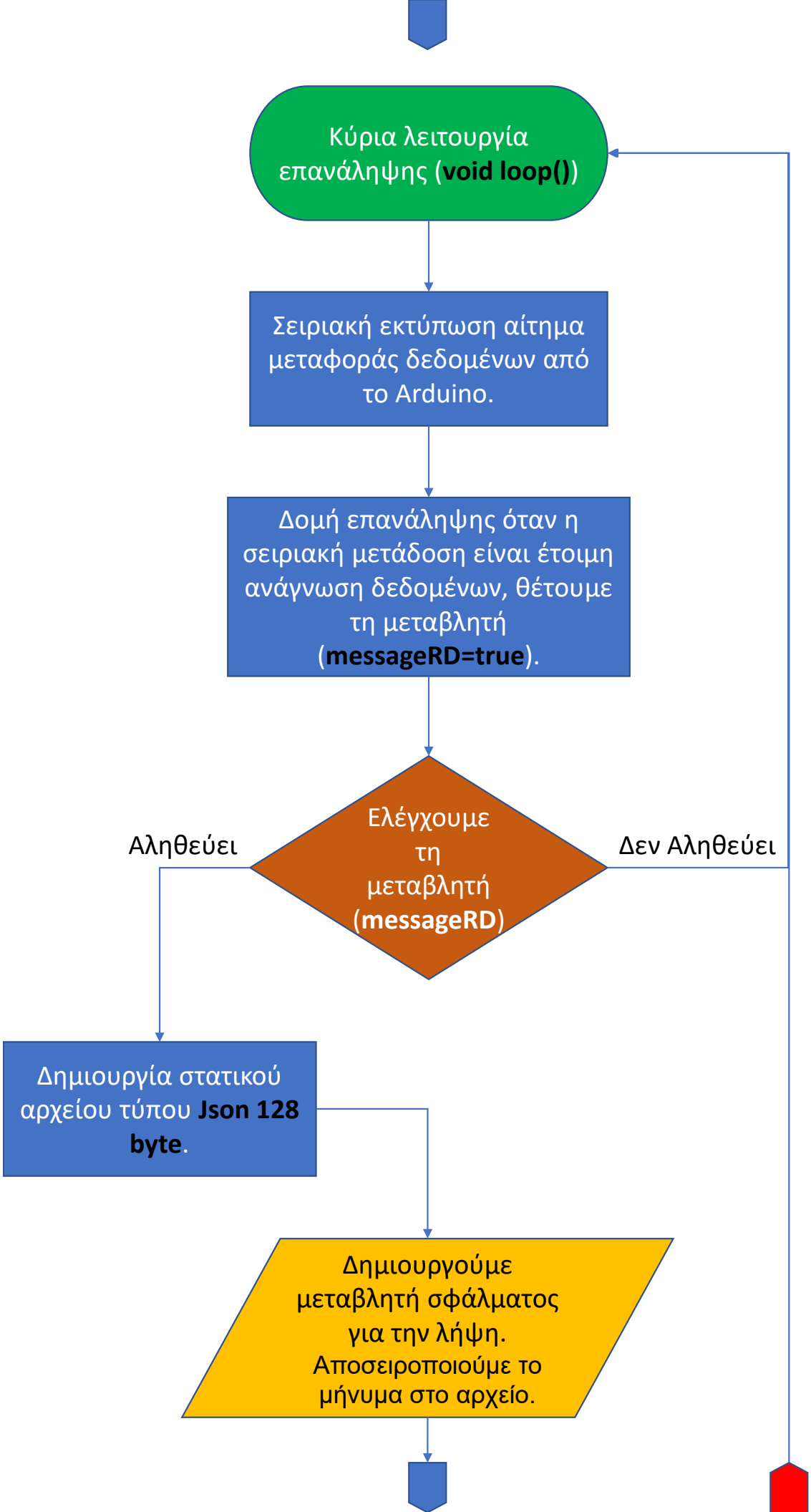
Ο Προγραμματισμός του ESP8266 δεν μπορεί να πραγματοποιηθεί απευθείας απο τον Η/Υ καθώς απαιτεί μια πλακέτα προγραμματισμού για να μπορέσουμε να θέσουμε το ESP8266 σε λειτουργία προγραμματισμού και να γράψουμε το πρόγραμμα στη μνήμη του. Έχοντας όμως την πλακέτα του Arduino μπορούμε να συνδέσουμε το ESP8266 απευθείας στους ακροδέκτες (RX, TX) της σειριακής επικοινωνίας καθώς αυτοί είναι συνδεδεμένοι μέσω της πλακέτας του Arduino στο ολοκληρωμένο που είναι υπεύθυνο για τον προγραμματισμό του, έτσι θέτοντας το ESP8266 σε λειτουργία προγραμματισμού θέτοντας τον ακροδέκτη FLASH σε λογικό ΄0΄ και κάνοντας επανεκκίνηση με τον RST ακροδέκτη το ESP8266 βρίσκετε σε λειτουργεία προγραμματισμού.

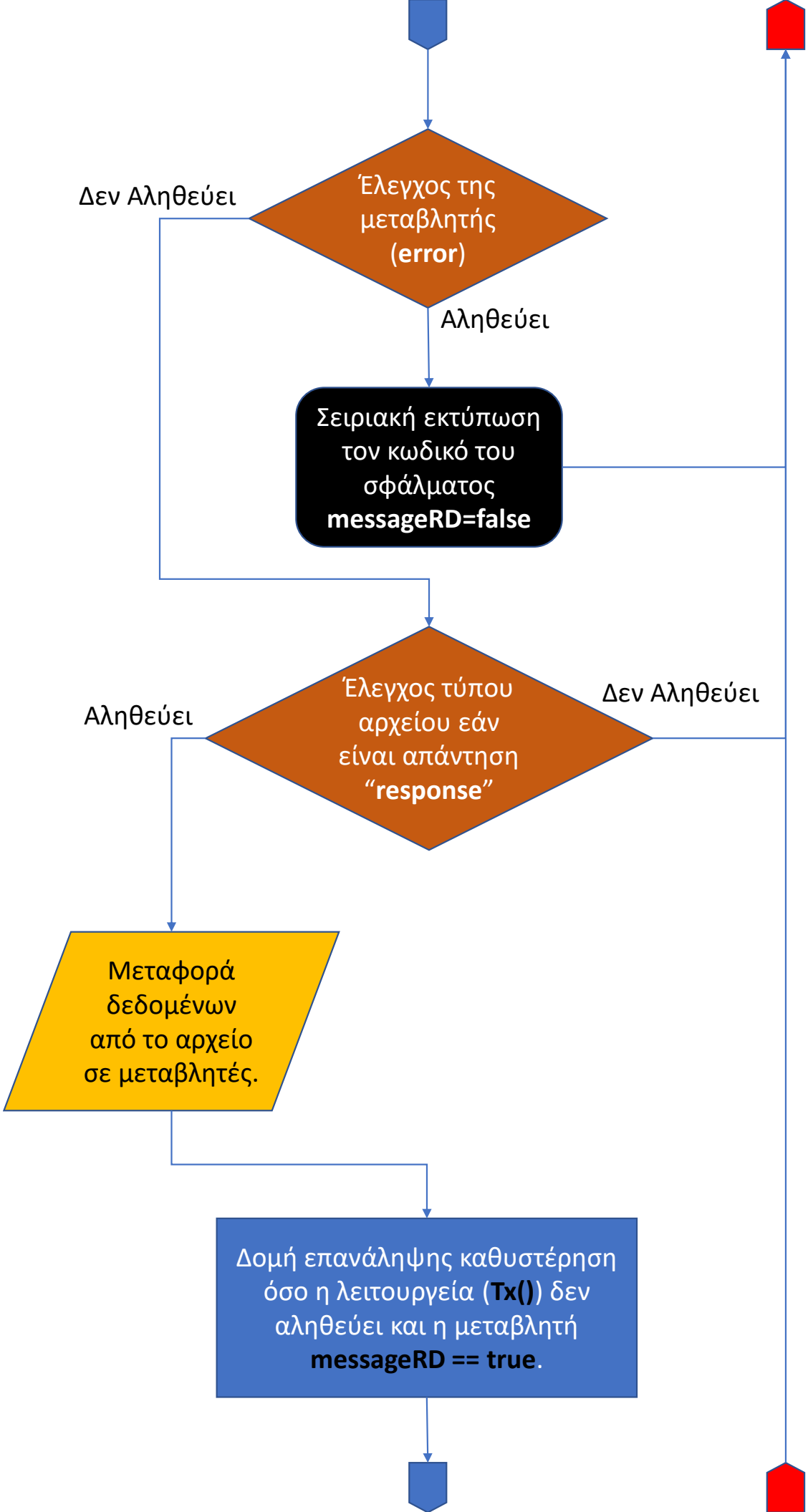
Από την μεριά του περιβάλλοντος προγραμματισμού θα πρέπει να το ρυθμίσουμε επιλέγοντας την πλακέτα που θέλουμε να προγραμματίσουμε πηγαίνοντας "Tools->Board:->Boards Manager" θα ανοίξει ένα παράθυρο αναζήτησης εκεί θα εισάγουμε την πλακέτα που θέλουμε να προγραμματίσουμε "ESP8266 Boards" και θα κάνουμε εγκατάσταση αυτού του πακέτου, αφού ολοκληρωθεί θα εμφανιστεί κάτω από το μενού Boards Manager η επιλογή "ESP8266 Boards(x.x.x)" ανοίγοντας το θα επιλέξομε "Generic ESP8266 Module".

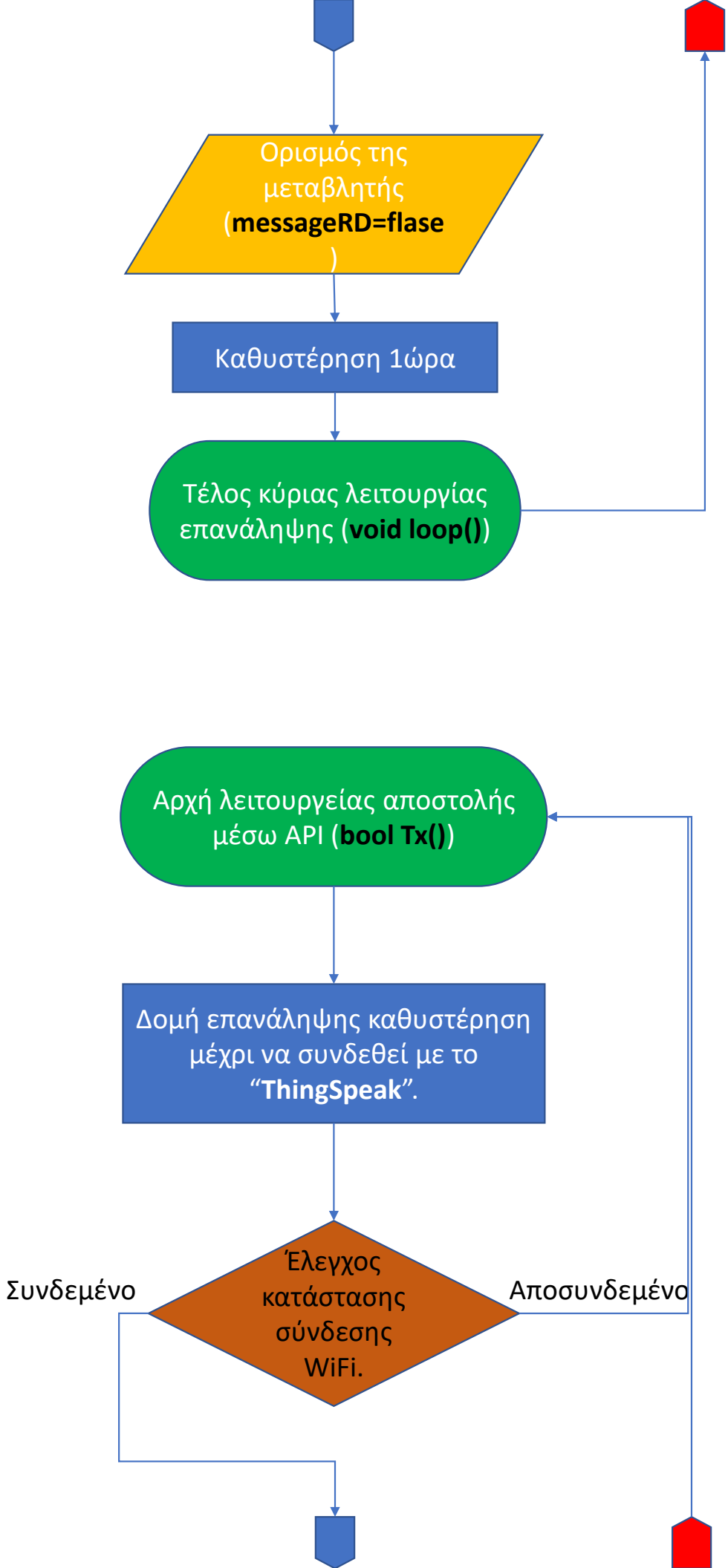
Τώρα το περιβάλλον προγραμματισμού μπορεί να μετατρέψει τον κώδικα του Arduino σε εντολές που μπορεί να καταλάβει το ESP8266.

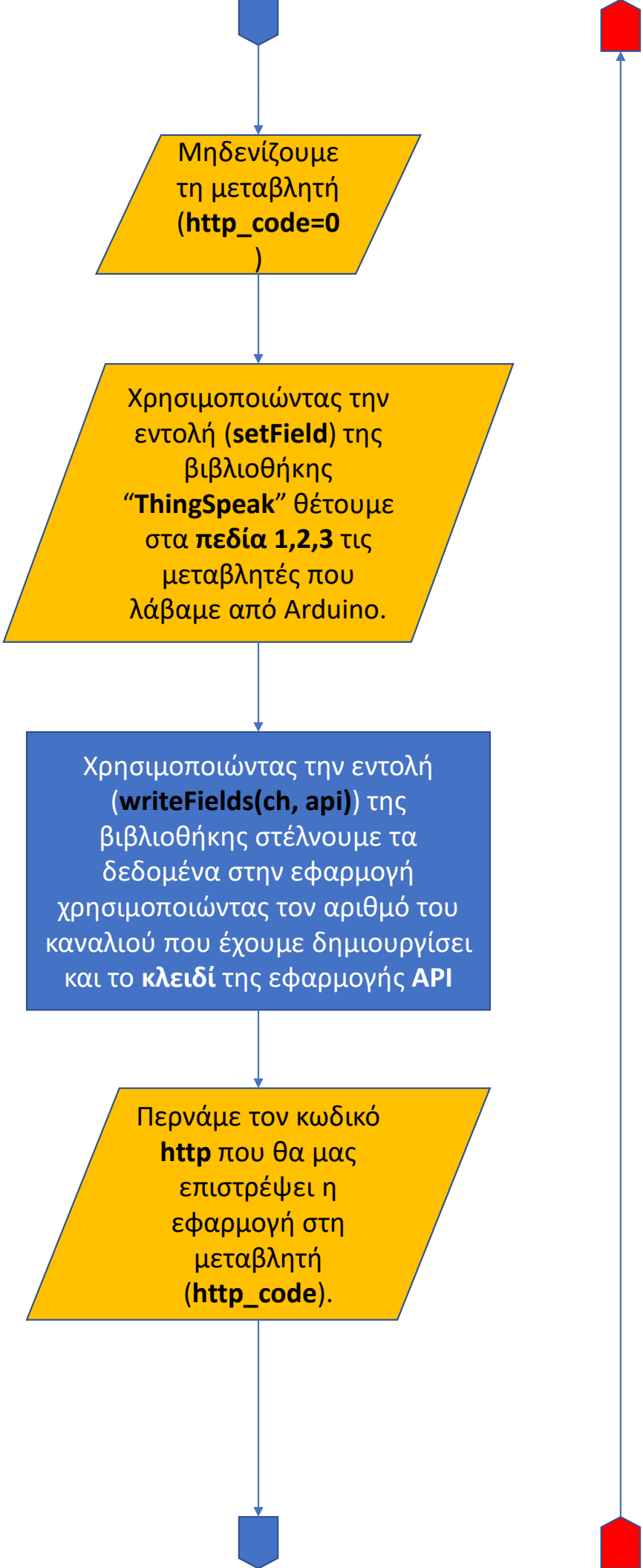
9.2.1 Κώδικας Flow Chart

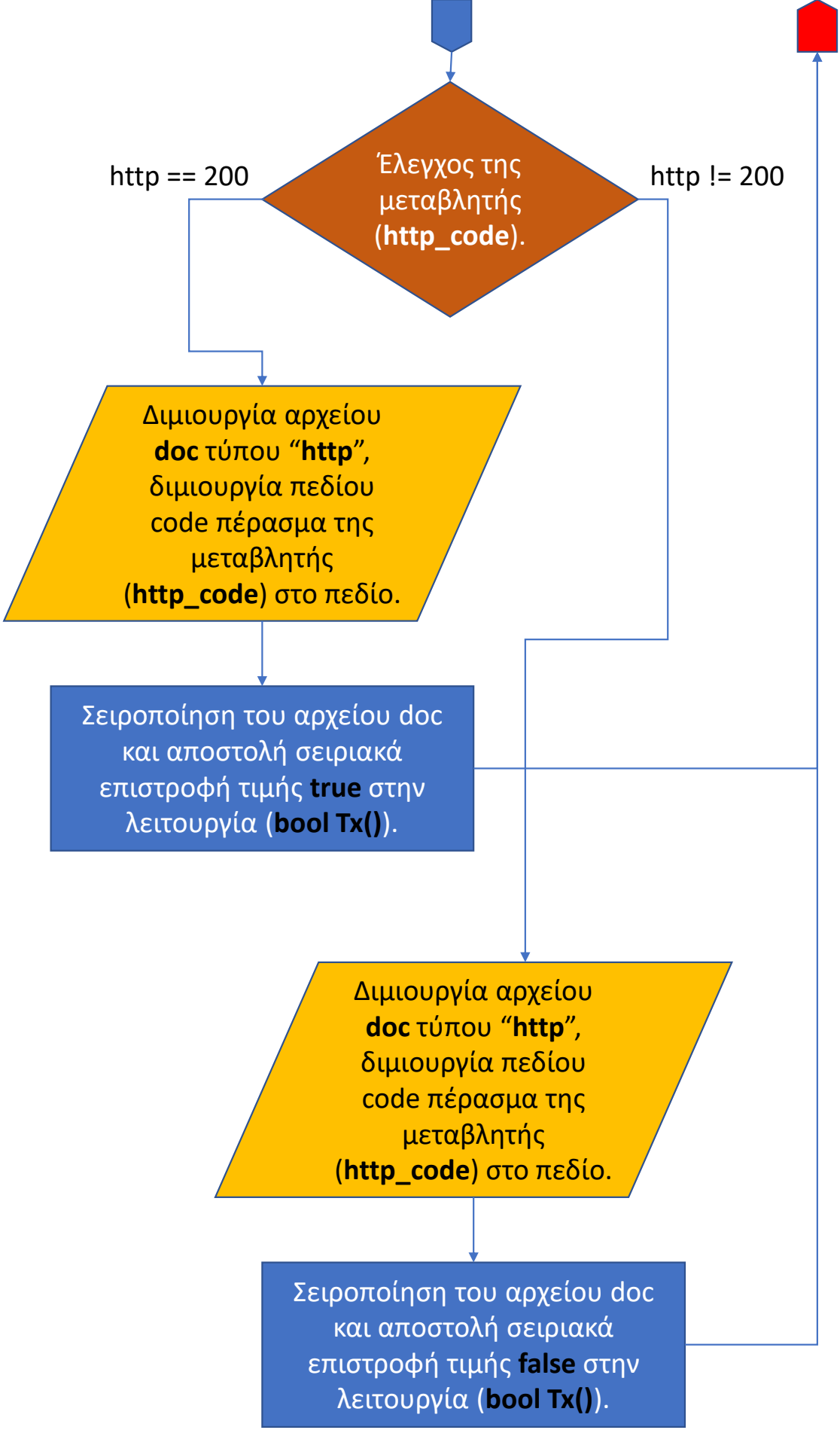

Εισαγωγή βιβλιοθηκών για την μεταγλώττιση του κώδικα στο ESP8266 καθώς και για την επικοινωνία με το ThingSpeak API.
Οριοθέτηση σταθερών τιμών και μεταβλητών, όπως το **ESSID** και τον **κωδικό σύνδεσης** με το δίκτυο καθώς και το **κλειδί** του API για την επικοινωνία με τον διακομιστή **ThingSpeak**
Εκκίνηση κύριας λειτουργίας ελέγχου (**void setup()**)
Εκκίνηση σειριακής επικοινωνίας, Θέτουμε το ESP8266 σε λειτουργία (**station mode**) χρήστη, Ξεκινάμε το WiFi και συνδεόμαστε στο δίκτυο που επιλέξαμε με τον κωδικό.
Δομή επανάληψης με καθυστέρηση μέχρι το ESP8266 να επιστρέψει (**status = WL_CONNECTED**).
Τέλος κύριας λειτουργίας ελέγχου (**void setup()**)


Κύρια λειτουργία επανάληψης (**void loop()**)
Σειριακή εκτύπωση αίτημα μεταφοράς δεδομένων από το Arduino.
Δομή επανάληψης όταν η σειριακή μετάδοση είναι έτοιμη ανάγνωση δεδομένων, θέτουμε τη μεταβλητή (**messageRD=true**).
Ελέγχουμε τη μεταβλητή (**messageRD**)
Αληθεύει
Δεν Αληθεύει
Δημιουργία στατικού αρχείου τύπου **Json 128 byte**.
Δημιουργούμε μεταβλητή σφάλματος για την λήψη. Αποσειροποιούμε το μήνυμα στο αρχείο.


Έλεγχος της μεταβλητής (error)
Δεν Αληθεύει
Αληθεύει
Σειριακή εκτύπωση τον κωδικό του σφάλματος messageRD=false
Έλεγχος τύπου αρχείου εάν είναι απάντηση "response"
Αληθεύει
Δεν Αληθεύει
Μεταφορά δεδομένων από το αρχείο σε μεταβλητές.
Δομή επανάληψης καθυστέρηση όσο η λειτουργεία (Tx()) δεν αληθεύει και η μεταβλητή messageRD == true.


Ορισμός της μεταβλητής (messageRD=flase)
Καθυστέρηση 1ώρα
Τέλος κύριας λειτουργίας επανάληψης (void loop())
Αρχή λειτουργείας αποστολής μέσω API (bool Tx())
Δομή επανάληψης καθυστέρηση μέχρι να συνδεθεί με το "ThingSpeak".
Έλεγχος κατάστασης σύνδεσης WiFi.
Συνδεμένο
Αποσυνδεμένο


Μηδενίζουμε τη μεταβλητή (**http_code=0**)
Χρησιμοποιώντας την εντολή (**setField**) της βιβλιοθήκης "**ThingSpeak**" θέτουμε στα **πεδία 1,2,3** τις μεταβλητές που λάβαμε από Arduino.
Χρησιμοποιώντας την εντολή (**writeFields(ch, api)**) της βιβλιοθήκης στέλνουμε τα δεδομένα στην εφαρμογή χρησιμοποιώντας τον αριθμό του καναλιού που έχουμε δημιουργίσει και το **κλειδί** της εφαρμογής **API**
Περνάμε τον κωδικό **http** που θα μας επιστρέψει η εφαρμογή στη μεταβλητή (**http_code**).


Έλεγχος της μεταβλητής (**http_code**).
http == 200
http != 200
Διμιουργία αρχείου **doc** τύπου "**http**", διμιουργία πεδίου code πέρασμα της μεταβλητής (**http_code**) στο πεδίο.
Σειροποίηση του αρχείου doc και αποστολή σειριακά επιστροφή τιμής **true** στην λειτουργία (**bool Tx()**).
Διμιουργία αρχείου **doc** τύπου "**http**", διμιουργία πεδίου code πέρασμα της μεταβλητής (**http_code**) στο πεδίο.
Σειροποίηση του αρχείου doc και αποστολή σειριακά επιστροφή τιμής **false** στην λειτουργία (**bool Tx()**).

9.2.2 Κώδικας Text

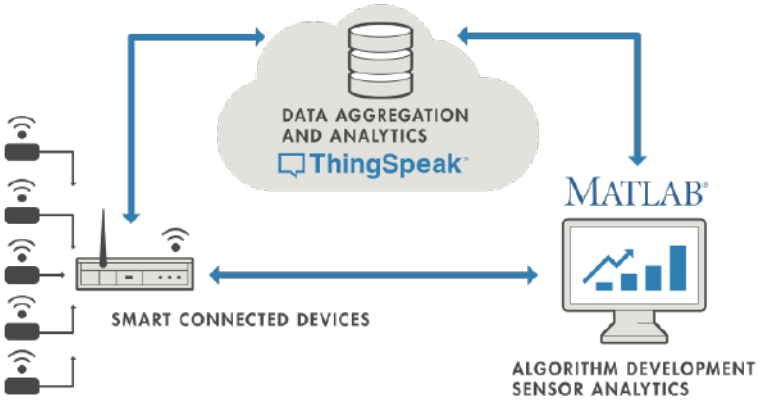
```
#include <ESP8266WiFi.h>//ESP8266 Library based on arduino WiFi//
#include <ThingSpeak.h>//Thing Speak library for HTTP communication
#include <ArduinoJson.h>//Arduino Json library for .Json data
transmission

//WiFi and Thing Speak parameters//
const char* ssid = "[redacted]";
const char* pass = "[redacted]";
const char* api = "[redacted]";
const long CH_ID = 1785753;

//Variables from station(Arduino UNO)//
int http_code;//HTTP return code i.e(200 OK)
String message = "";
bool messageRD = false;

//input data types//
char tmp = 0;//temperature value type
long pres = 0;//pressure value type
byte hum = 0;//humidity value type

WiFiClient client;

void setup() {

  Serial.begin(9600);
  WiFi.mode(WIFI_STA);//Set ESP8266 in station mode
  WiFi.begin(ssid, pass);//Connect to an Access Point declared in
variables

  //Wait for the esp to connect to the AP//
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    //Serial.print(".");
  }

  //Get local IP//
  //Serial.println("Connected");
  //Serial.print("IP: ");
  //Serial.print(WiFi.localIP());
}

void loop() {

  Serial.println("{\"type\":\"request\"}");//print the Json request

  while (Serial.available())
```

```
  {
    message = Serial.readString();//read the serialized Json data
    messageRD = true;
  }

  if (messageRD)//when the data is received
  {
    //Set Json document type and size//
    StaticJsonDocument<128> doc;

    //Deserialize Json data and return error code//
    DeserializationError error = deserializeJson(doc, message);

    if (error)//if an error code is returned print it
    {
      //Serial.print("ERROR: ");
      //Serial.println(error);
      messageRD = false;//no message received
      return;
    }

    //if the message type is response fill the variables with the
fields of the Json file//
    if (doc["type"] == "response")
    {
      tmp = doc["tmp"];
      hum = doc["hum"];
      pres = doc["pres"];
    }

    while (Tx() != true && messageRD == true) //call the thingspeak
function to send the data
    {
      delay(100);
    }
    messageRD = false;//reset the message indicator for the next
transaction
  }
  delay(3600000);//wait 1 hour
}

//ThinkSpeak Send Function//
bool Tx()
{
  while (!ThingSpeak.begin(client)) //wait until it connects to
thingspeak.com api
  {
    delay(100);
```

```
  }

  if (WiFi.status() == WL_CONNECTED)
  {
    http_code = 0;//zero the http code variable

    //Move values from the variables to the ThingSpeak fields//
    ThingSpeak.setField(1, tmp);
    ThingSpeak.setField(2, pres);
    ThingSpeak.setField(3, hum);

    //POST values, command to the specified chanel ID with the api key
required//
    http_code = ThingSpeak.writeFields(CH_ID, api);

    if (http_code == 200)//HTTP 200 means OK
    {
      //Serial.print("Channel updated HTTP/1.1: ");
      //Serial.println(http_code);
      //test code for debuging ^^

      doc["type"] = ["http"];
      doc["code"] = http_code;
      serializeJson(doc, Serial);
      //respond the http code//

      return true;
    }

    else//if not 200 print the code response code
    {
      //Serial.print("Problem updating HTTP/1.1: ");
      //Serial.println(http_code);
      //test code for debuging when the module programed^^

      doc["type"] = ["http"];
      doc["code"] = http_code;
      serializeJson(doc, Serial);
      //respond the http code//

      return false;
    }

  }
}
```

10 Δημιουργία Εφαρμογής ThingSpeak

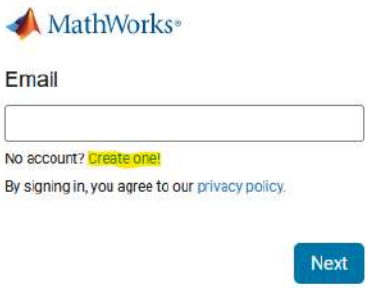
Το ThingSpeak είναι ένα λογισμικό ανοιχτού κώδικα γραμμένο στη γλώσσα προγραμματισμού Ruby το οποίο επιτρέπει στους χρήστες να επικοινωνούν με συσκευές με την δυνατότητα διαδικτύου. Παρακάτω θα περιγράψουμε τα βήματα που ακολουθήθηκαν για την δημιουργία της εφαρμογής.



Σχήμα 10: Αρχιτεκτονική δικτύου ThingSpeak

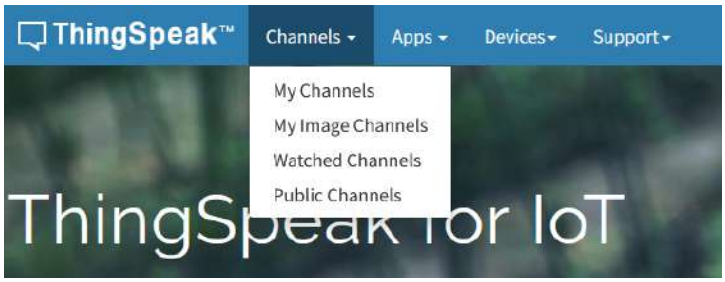
10.1 Βήμα 1ο Δημιουργία Λογαριασμού

Επισκέπτοντας το την σελίδα του
ThingSpeak Sign In
Κάτω από το πεδίο εισόδου του Email επιλέγουμε την δημιουργία λογαριασμού.



Σχήμα 11: ThingSpeak Βήμα 1ο

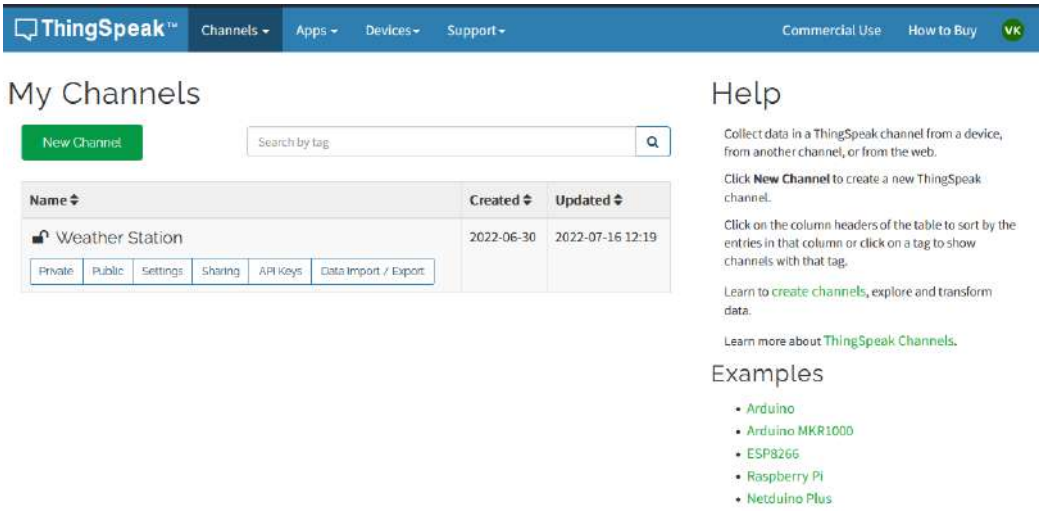
Έπειτα αφού συνδεθούμε με τον λογαριασμό μας, πρέπει να ανοίξουμε ένα νέο κανάλι για την συλλογή των δεδομένων μας.
Από την αρχική σελίδα πηγαίνουμε στο υπομενού "Channels -> My Channels"



Σχήμα 12: ThingSpeak Βήμα 1.1

10.2 Βήμα 2ο Δημιουργία Καναλιού

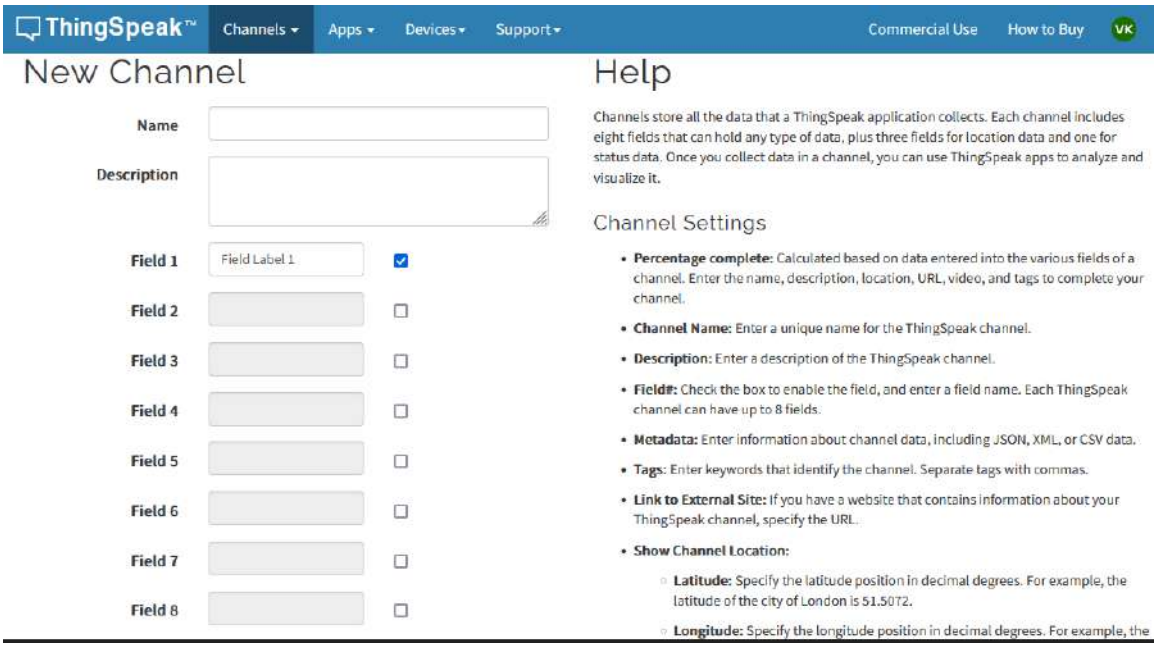
Πηγαίνοντας στην επιλογή "My Channels" θα μας ανοίξει μία σελίδα με τα υπάρχοντα κανάλια και μία επιλογή για την δημιουργία νέου καναλιού "New Channel" (το υπάρχον κανάλι όπως στην φωτογραφία παρακάτω δεν θα υπάρχει εφόσον το δημιουργούμε πρώτη φορά).



Σχήμα 13: ThingSpeak Βήμα 2ο

10.3 Βήμα 3ο Συμπλήρωση Πεδίων

Αφού πατήσουμε την επιλογή για την δημιουργία νέου καναλιού θα μας ανοίξει μία νέα σελίδα για να συμπληρώσουμε δεδομένα σχετικά με το κανάλι επικοινωνίας.
Το πρώτο πεδίο που συναντάμε είναι το όνομα που θέλουμε να δώσουμε στο κανάλι μας, μετά ακολουθεί μία σύντομη περιγραφή.
Στη συνέχεια μας δίνεται η επιλογή να επιλέξουμε τον αριθμό των πεδίων που θέλουμε, (κάθε πεδίο λαμβάνει απο μία τιμή) και το ονομάζουμε, π.χ ¨Θερμοκρασία¨.



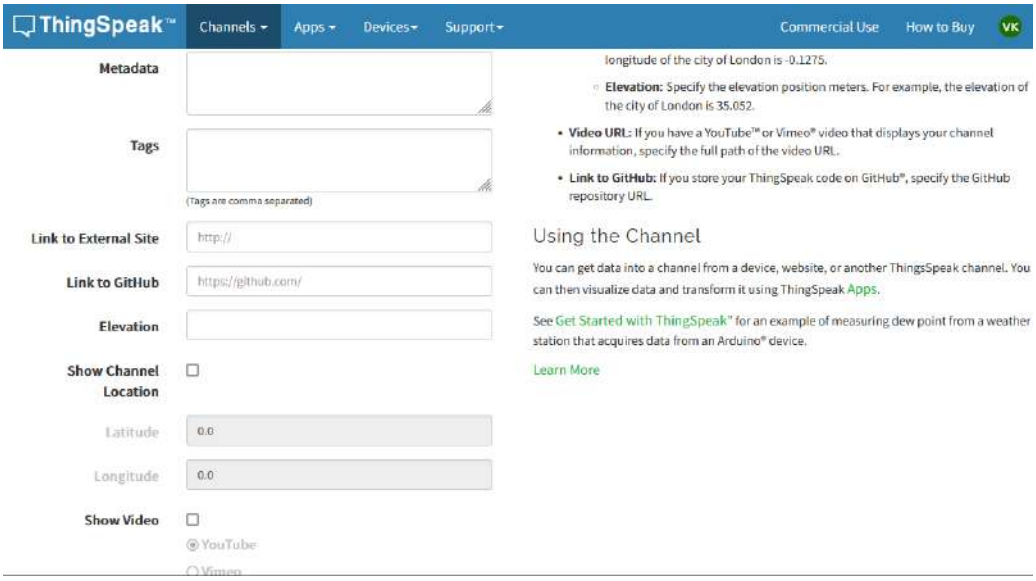
Σχήμα 14: ThingSpeak Βήμα 3ο

Παράλληλα μπορούμε να δούμε στο δεξί μέρος της σελίδας μας δίνετε μία περιγραφή για το τι σημασία έχει το κάθε ένα πεδίο.

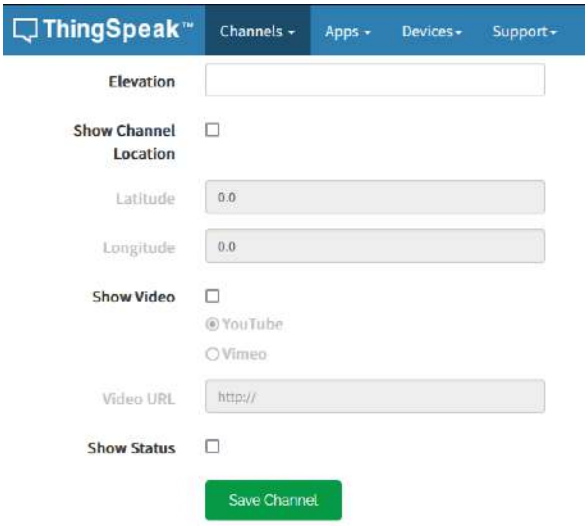
Παρακάτω με τη σειρά βλέπουμε τα πεδία:

- Metadata
- Tags
- Link to external site
- Link to GitHub
- Elevation
- Show Channel Location
- Geographic Location (Lat,Long)

Από τα πεδία που προαναφέρθηκαν αυτά που μας ενδιαφέρουν περισσότερο είναι τα: Υψόμετρο, Γεωγραφική τοποθεσία (εαν το επιθυμούμε), Tags, Metadata.



Σχήμα 15: ThingSpeak Βήμα 3.1

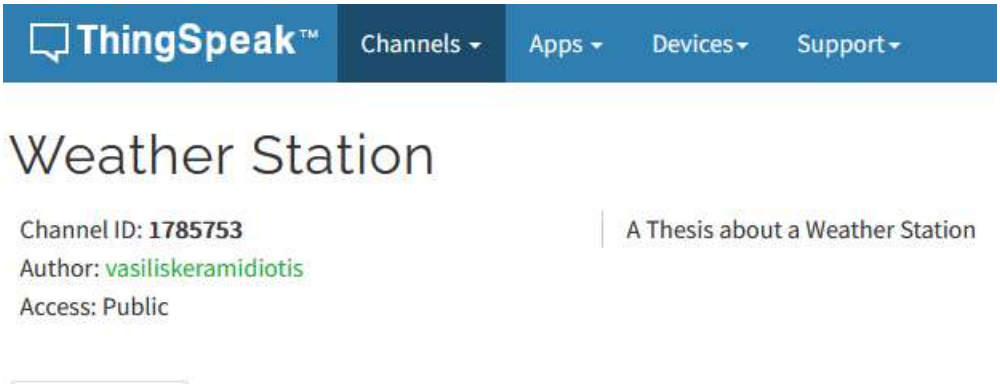


Σχήμα 16: ThingSpeak Βήμα 3.2

Τέλος αφού συμπληρώσουμε τα πεδία που θέλουμε πατάμε αποθήκευση καναλιού εμφανίζοντας το κανάλι μας στη λίστα "My Channels".

10.4 Βήμα 4ο Αντιγραφή Κλειδιών API

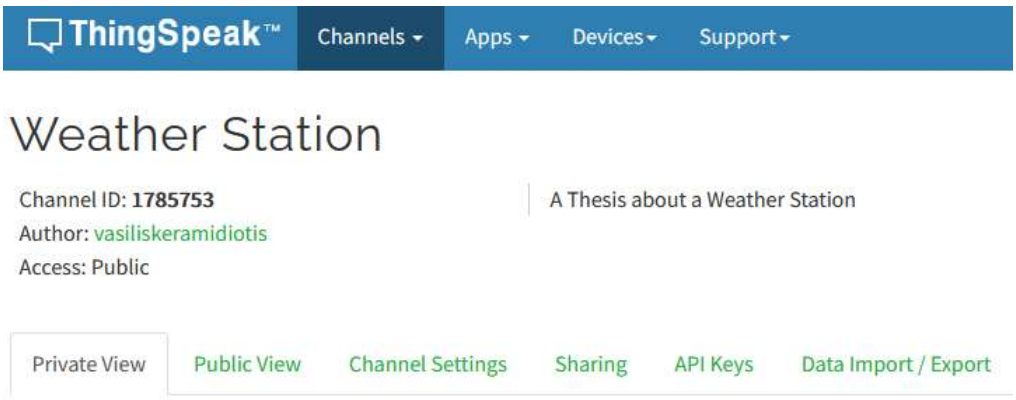
Ανοίγοντας το κανάλι που δημιουργίσαμε μας ανοίγει η κύρια σελίδα όπου περιγράφει το όνομα του καναλιού, την περιγραφή, τον αριθμό του καναλιού Channel ID, το όνομα του δημιουργού και την πρόσβαση του καναλιού δηλαδή αν είναι δημόσιο ή ιδιωτικό.



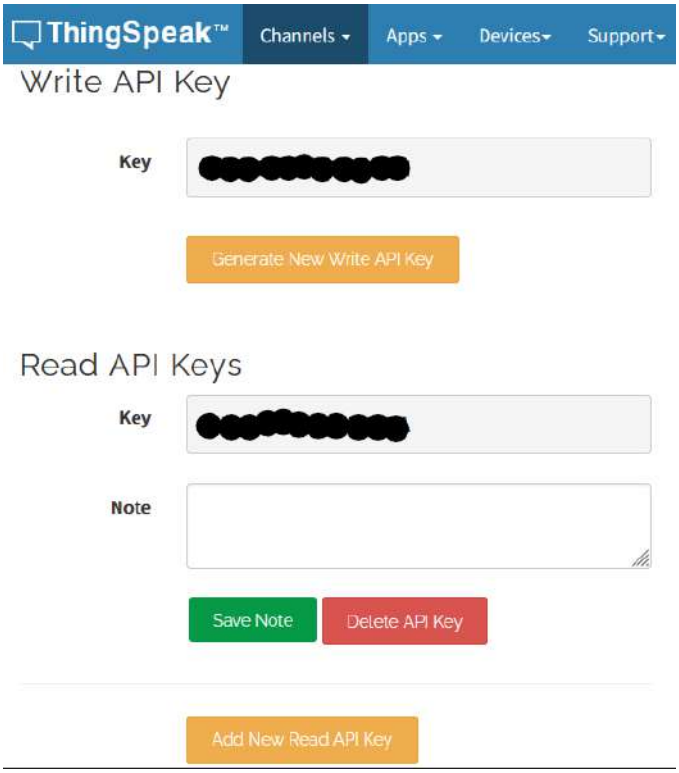
Σχήμα 17: ThingSpeak Βήμα 4ο

Παρακάτω βλέπουμε κάποιες καρτέλες:

- Private View: Η όψη των δεδομένων που θα βλέπουμε μόνο εμείς (ο ιδιοκτήτης).
- Public View: Η όψη των δεδομένων που θα βλέπουν οι υπόλοιποι χρήστες.
- Channel Settings: Οι ρυθμήσεις του καναλιού (η ίδια σελίδα για την δημιουργία).
- Sharing: Διαμοιρασμός του καναλιού με όλους, με επιλεγμένους χρήστες ή καθόλου.
- API Keys: Τα κλειδιά της εφαρμογής για την ταυτοποίηση και την μετάδοση των δεδομένων.
- Data Import Export: Εισαγωγή ή Εξαγωγή δεδομένων σε διάφορες μορφές αρχείων.



Σχήμα 18: ThingSpeak Βήμα 4.1

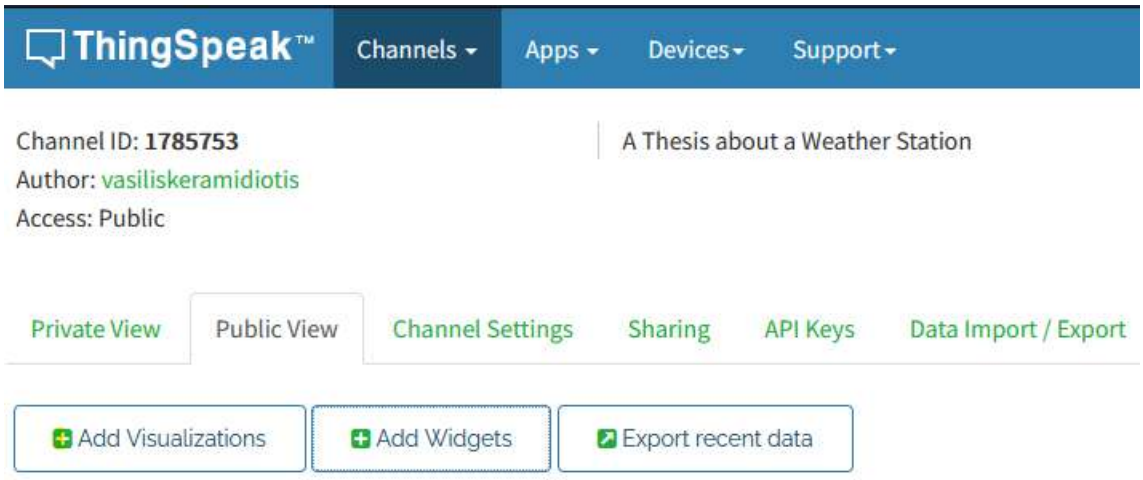


Σχήμα 19: ThingSpeak Βήμα 4.2

Η Καρτέλα που μας ενδιαφέρει πρώτα είναι η API Keys ώστε να αντιγράψουμε το κλειδί για την αποστολή δεδομένων (Write API Key) στο κανάλι και να το περάσουμε στον κώδικα ώστε να μπορεί να συνδεθεί και να στείλει τα δεδομένα.

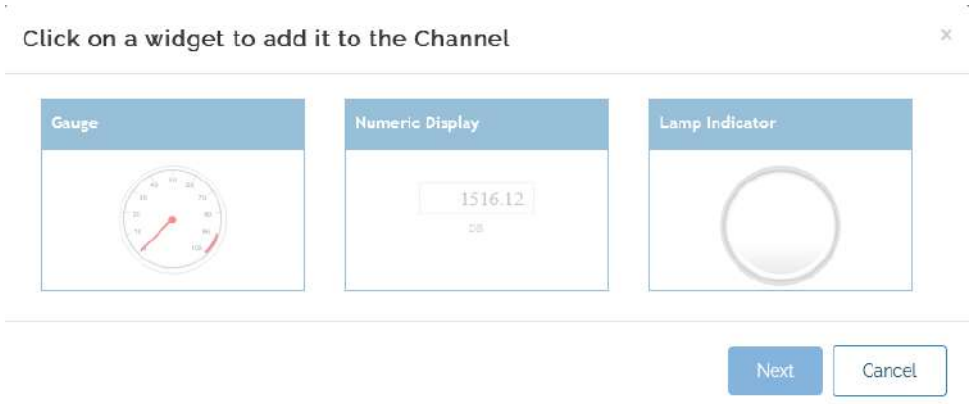
10.5 Βήμα 5ο Εισαγωγή Γραφημάτων και Οργάνων

Στις καρτέλες Private - Public view αντίστοιχα στη κάθε μια μπορούμε να εισάγουμε είτε στοιχεία όπως όργανα ένδειξης, λυχνίες και πεδία τιμών (Add Widgets), είτε γραφήματα σχετικά με το κάθε πεδίο τιμών που συλλέγουμε (Add Visualizations).



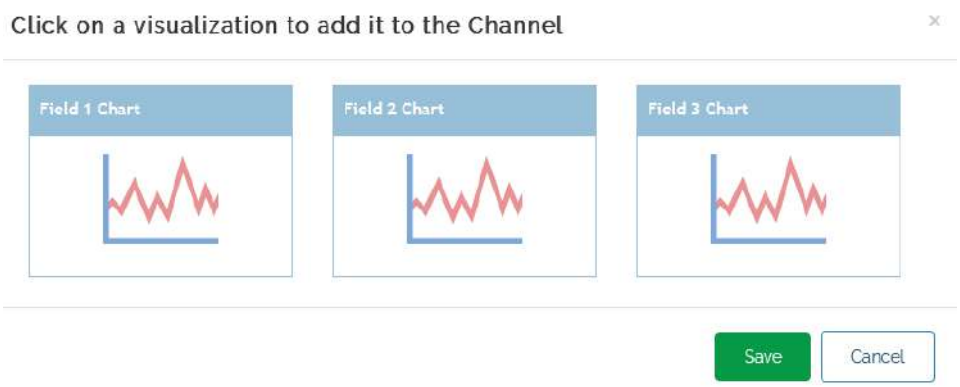
Σχήμα 20: ThingSpeak Βήμα 5ο

Όργανα ένδειξης:



Σχήμα 21: ThingSpeak Όργανα ένδειξης

Γραφήματα:

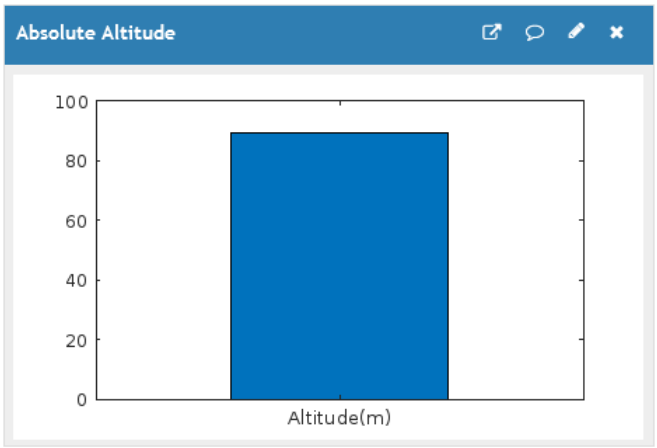


Σχήμα 22: ThingSpeak Γραφήματα

Παράδειγμα οργάνων:

Σχήμα 23: ThingSpeak Παράδειγμα οργάνων

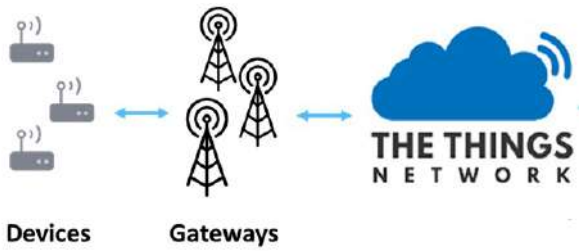
Παράδειγμα Γραφημάτων:



Σχήμα 24: ThingSpeak Παράδειγμα γραφημάτων

11 Δημιουργία Εφαρμογής The Things Network

Το The Things Network είναι ένα παγκόσμιο συνεργατικό Internet of Things οικοσύστημα το οποίο μας επιτρέπει να δημιουργήσουμε και να συνδέσουμε μεταξύ τους δίκτυα και συσκευές, χρησιμοποιώντας το πρωτόκολλο LoRa WAN.

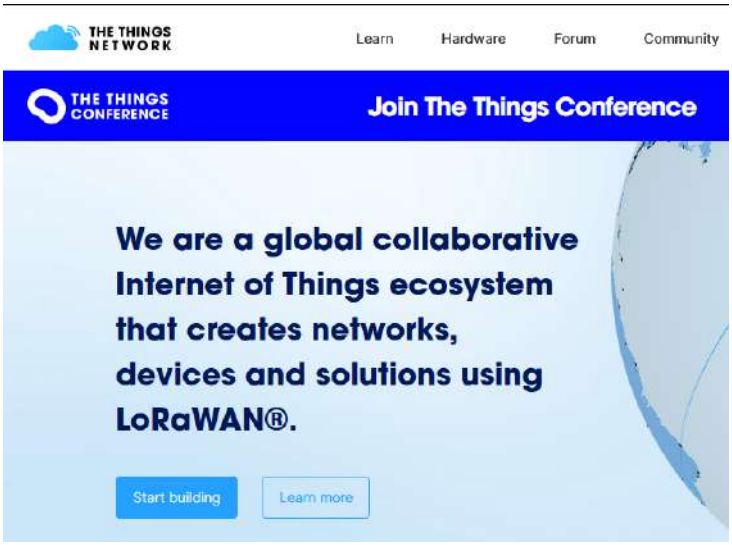


Σχήμα 25: Δίκτυο TTN

11.1 Βήμα 1ο Δημιουργία λογαριασμού

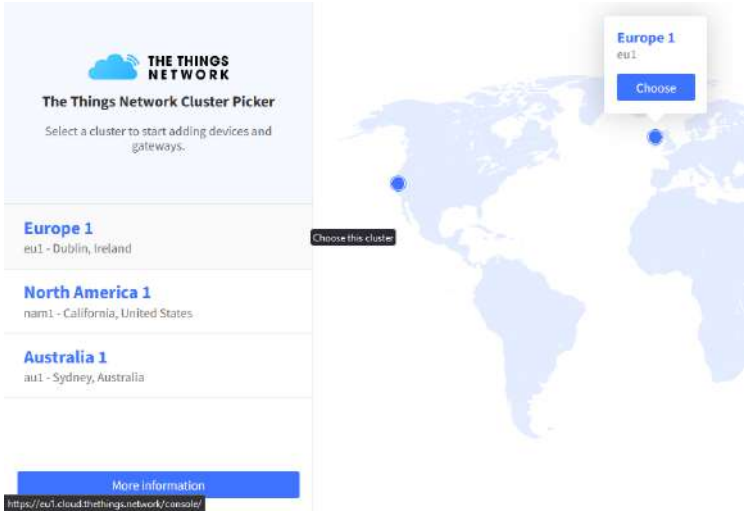
Για την δημιουργία λογαριασμού στο The Things Network αφού έρθουμε στην αρχική σελίδα επιλέγουμε επάνω δεξιά Sing Up και ακολουθούμε τα βήματα για την εγγραφή ή επιλέγουμε τον σύνδεσμο παρακάτω για την δημιουργία λογαριασμού στο Ευροπαϊκό Δίκτυο.
The Things Network Sing Up

Αφού συνδεθούμε επιλέγουμε το πεδίο Start building για να ξεκινήσουμε την δημιουργία της εφαρμογής τελικής συσκευής LoRaWAN.



Σχήμα 26: TTN Βήμα 1ο

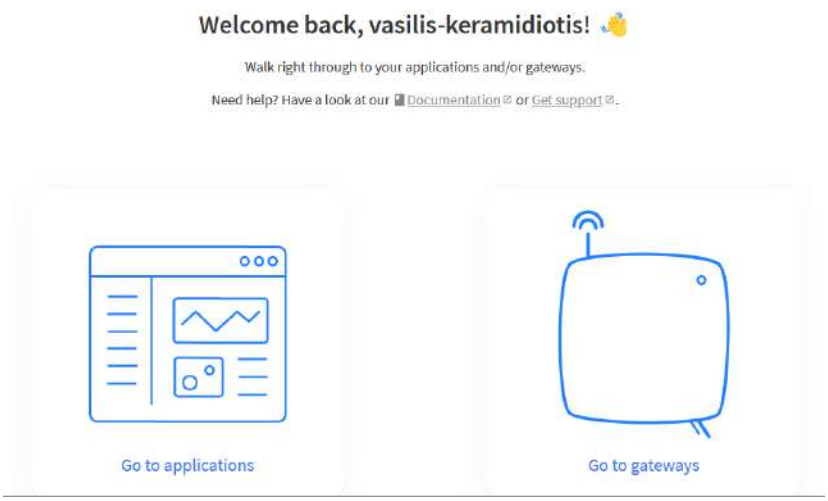
Στη συνέχεια θα μας ζητηθεί σε ποιο δίκτυο θέλουμε να δημιουργίσουμε την εφαρμογή μας, επιλέγοντας Ευρώπη συνεχίζουμε.



Σχήμα 27: TTN Βήμα 1.1

11.2 Βήμα 2ο Δημιουργία εφαρμογής

Εφόσον βρισκόμαστε στην κεντρική σελίδα επιλέγουμε να κατευθυνθούμε στην καρτέλα με τις εφαρμογές μας "Go to applications".

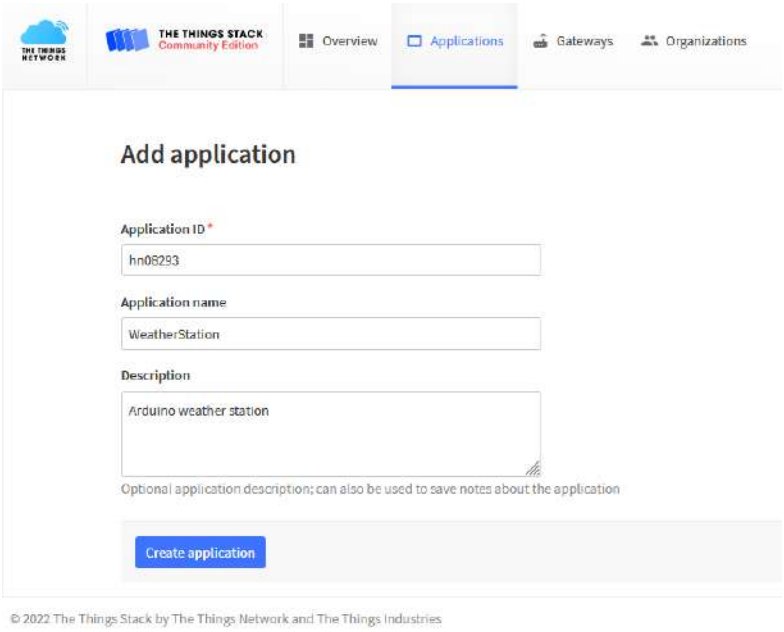


Σχήμα 28: TTN Βήμα 2ο

Στη συνέχεια επιλέγουμε στα δεξιά της σελίδας να προσθέσουμε εφαρμογή "Add application".

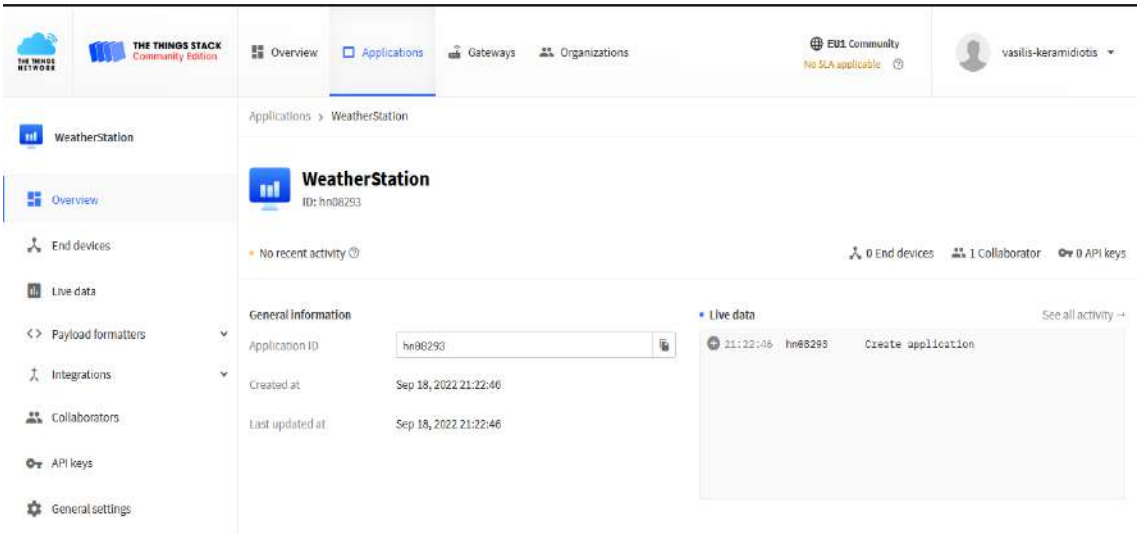
Σχήμα 29: TTN Βήμα 2.1

Στα πεδία που θα εμφανιστούν καλούμαστε να συμπληρώσουμε έναν αναγνωριστικό κωδικό για τη εφαρμογή μας καθώς και μια ονομασία, προαιρετικά μπορούμε να δώσουμε και μία περιγραφή.



Σχήμα 30: TTN Βήμα 2.2

Αφού ολοκληρώσουμε το προηγούμενο βήμα θα μας εμφανιστεί μια περιληπτική περιγραφή της εφαρμογής μας. Στην αριστερή πλευρά της σελίδας υπάρχουν κάποιες καρτέλες περιήγησης, αυτή που μας ενδιαφέρει στο επόμενο βήμα είναι η καρτέλα των συσκευών "End Devices"που έχουμε καταχωρίσει στην εφαρμογή μας.



Σχήμα 31: TTN Βήμα 2.3

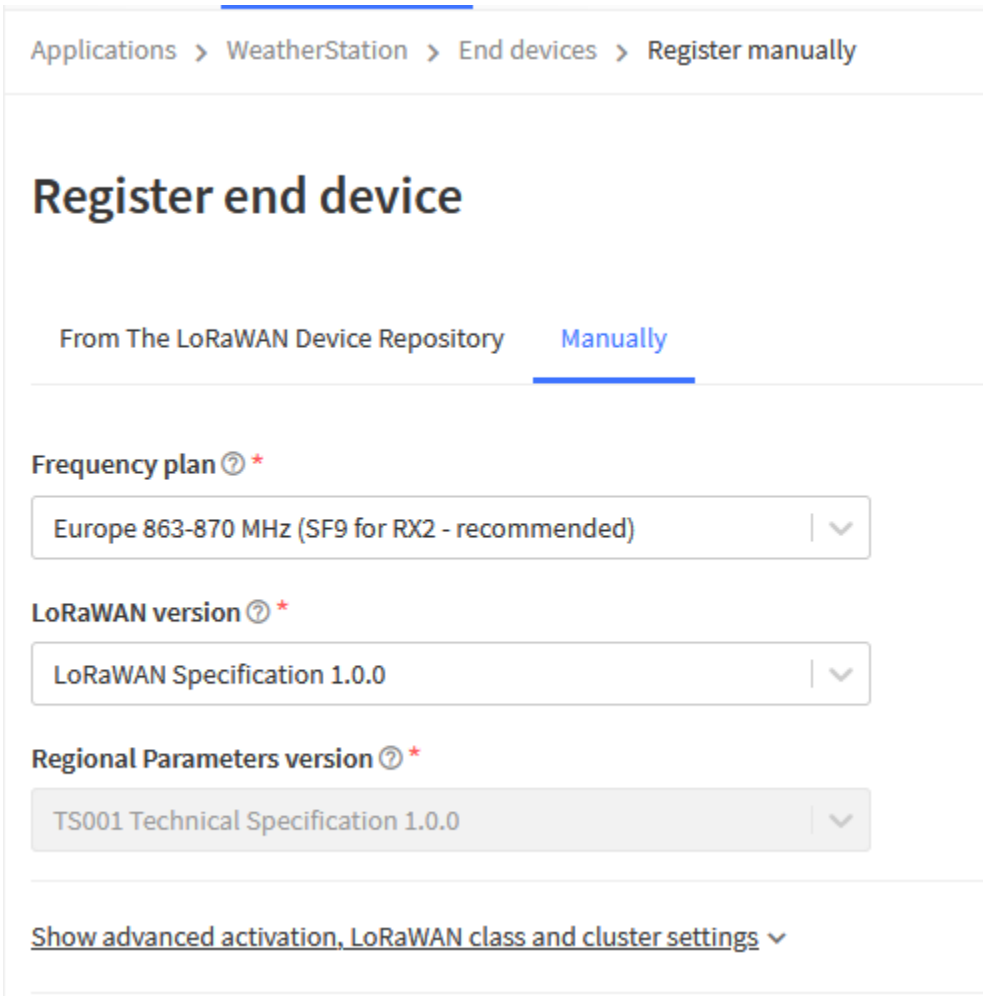
Εκεί θα επιλέξουμε να προσθέσουμε νέα συσκευή.

Σχήμα 32: TTN Βήμα 2.4

11.3 Βήμα 3ο Καταχώριση συσκευής

Για να καταχωρίσουμε την συσκευή μας θα επιλέξουμε την χειροκίνητη μέθοδο καθώς δεν είναι έτοιμη για απευθείας σύνδεση διότι δεν υπάρχει στην λίστα συσκευών, αλλά είναι μία συσκευή που έχει σχεδιασθεί από εμάς.

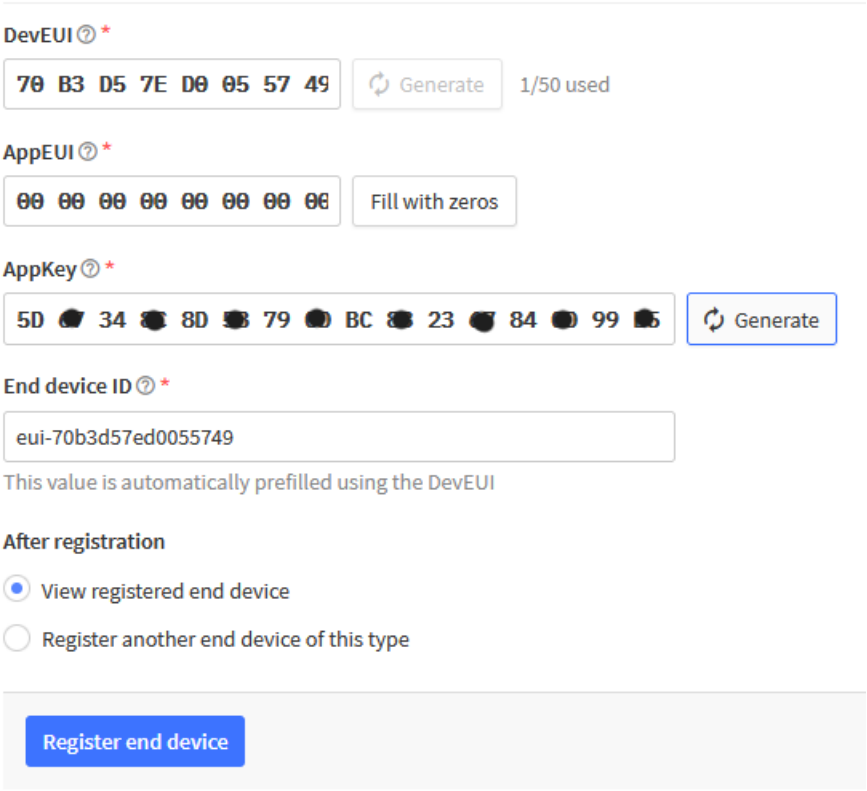
Στα πεδία που εμφανίζονται θα επιλέξουμε το σχέδιο συχνότητας που θα χρησιμοποιήσουμε, την έκδοση του πρωτοκόλλου τα οποία καθορίζονται από τον πομποδέκτη που έχουμε επιλέξει.



Σχήμα 33: TTN Βήμα 3ο

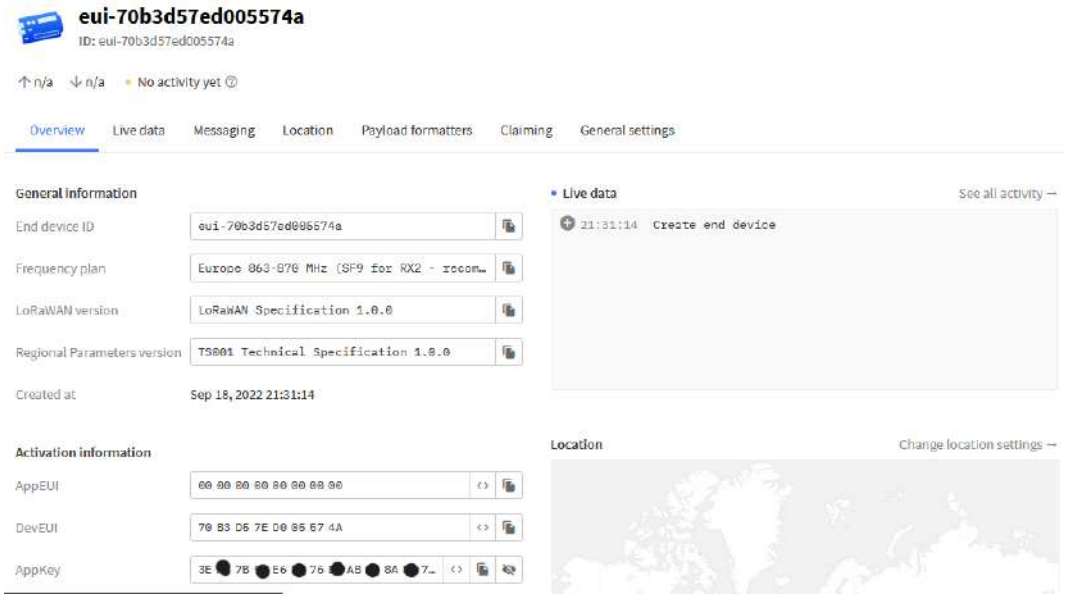
Συνεχίζοντας παρακάτω επιλέγουμε να παράξουμε τα: DevEUI - AppEUI - AppKey τα οποία χρειάζονται για την σύνδεση και αποδοχής της συσκευής μας.

- DevEUI: 64bit Μοναδικό αναγνωριστικό συσκευής.
- AppEUI: 64bit Μοναδικό αναγνωριστικό εφαρμογής απαραίτητο για την χειραψία κατά την σύνδεση στην εφαρμογή.
- AppKey: AES-128 bit Μοναδικό και κρυφό κλειδί απαραίτητο για την κρυπτογράφηση των δεδομένων πριν την αποστολή, αυτό το κλειδί δεν μεταδίδεται ποτέ στο δίκτυο.



Σχήμα 34: TTN Βήμα 3.1

Αφού ολοκληρώσουμε τα βήματα και επιλέξουμε την καταχώριση της συσκευής θα επιστρέψουμε στην σελίδα που περιγράφει την συσκευή μας.



Σχήμα 35: Βήμα 3.2

Τέλος αντιγράφοντας αυτά τα τρία πεδία των αναγνωριστικών συσκευής και εφαρμογής καθώς και το AES κλειδί μπορούμε να ενεργοποιήσουμε την συσκευή μας χρησιμοποιώντας τον κώδικα ttn-otaa(the things network-Over The Air Activation) και ακολουθώντας τις οδηγίες της βιβλιοθήκης MCCI-LoRaWAN-LMIC-library στο GitHub0 αντιγράφουμε στο πρόγραμμα του Arduino τις τιμές αυτές.

11.4 OTAA (Over the Air Activation)

Η διαδικασία [9] Over the Air Activation ή ενεργοποίηση μέσω του αέρα είναι μια ασφαλής διαδικασία για να ενεργοποιήσουμε και να προσθέσουμε μια συσκευή στο δίκτυο LoRaWAN.
Η διαδικασία αυτή αποτελείτε από μία ακολουθία 5 βημάτων.

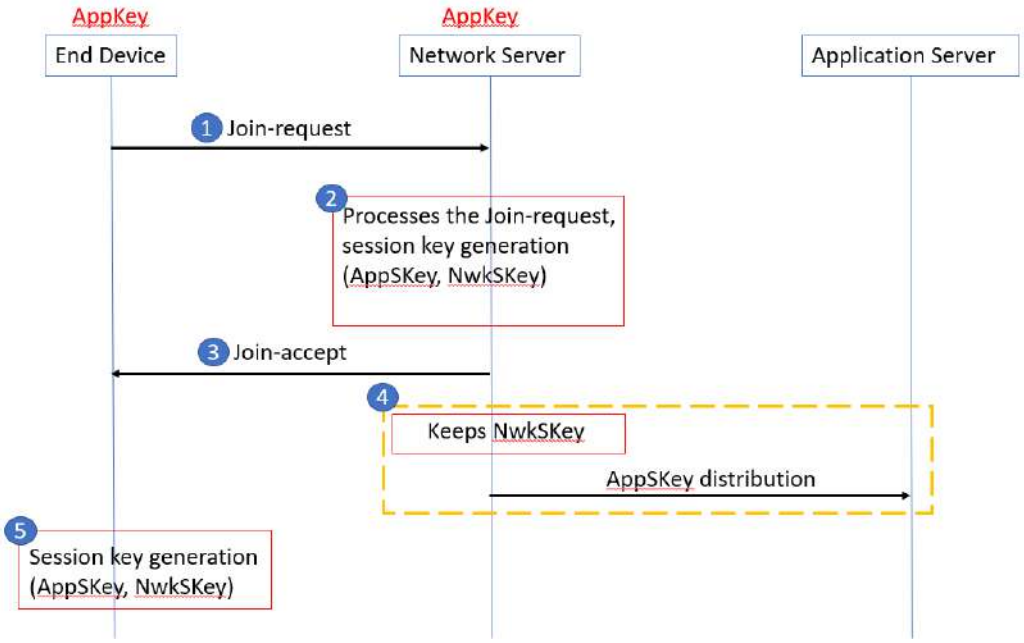
- Βήμα 1: Η συσκευή μας ξεκινά την αποστολή του μηνύματος σύνδεσης στο δίκτυο, το μήνυμα αυτό αποτελείτε από τρία πεδία, δύο από αυτά τα προαναφέραμε παραπάνω DevEUI - AppEUI και ένα ακόμη το DevNonce το οποίο είναι μια μοναδική τυχαία τιμή των $2byte$ η οποία δημιουργείτε από την συσκευή μας και αποθηκεύεται από τον εκάστοτε διακομιστή του δικτύου και μπορεί να διακρίνει πια συσκευή έστειλε το μήνυμα σύνδεσης, με αποτέλεσμα να απορρίψει άλλα μηνύματα σύνδεσης με την ίδια τιμή προστατεύοντας έτσι το δίκτυο από επιθέσεις αναμετάδοσης (replay attack).
- Βήμα 2: Ο διακομιστής αφού δεχθεί το μήνυμα σύνδεσης το επεξεργάζεται και παράγει δύο κλειδιά συνεδρίας (session keys) τα (NwkSKey - AppSKey) καθώς και το μήνυμα αποδοχής εάν η συσκευή επιτρέπετε να εισέλθει στο δίκτυο.

  Το μήνυμα αποδοχής περιλαμβάνει:

  - AppNonce: Είναι μια τυχαία τιμή που δημιουργείτε από τον διακομιστή και χρησιμοποιήτε από την συσκευή για να εξάγει τα κλειδιά (NwkSKey - AppSKey).
  - NetID: Περιέχει το αναγνωριστικό του δικτύου, τα 7 περισσότερο σημαντικά ψηφία ή MSB.
  - DevAddr: Είναι η διεύθυνση της συσκευής την οποία παράγει ο διακομιστής για να αναγνωρίζει την συσκευή στο παρών δίκτυο και αποτελείτε από $32bit$.
  - DLSettings: Αυτό το πεδίο αποτελείτε από $1Byte$ και περιέχει τις ρυθμίσεις κατερχόμενης σύνδεσης που πρέπει να χρησιμοποιήσει η συσκευή μας.
  - RxDelay: Περιέχει τον χρόνο καθυστέρησης που θα χρησιμοποιήσει η συσκευή μεταξύ της αποστολής και λήψης δεδομένων Rx - Tx.
  - CFList: Προαιρετική λίστα συχνοτήτων που υποστηρίζει ο διακομιστής.

    Όλο αυτό το μήνυμα κρυπτογραφήτε με το (AppKey) χρησιμοποιώντας κρυπτογράφηση AES.

- Βήμα 3: Ο διακομιστής στέλνει στη συσκευή το κρυπτοφραφημένο μήνυμα αποδοχής.
- Βήμα 4: Ο διακομιστής του δικτύου κρατά το κλειδί δικτύου (NwkKey) και μεταφέρει το κλειδί εφαρμογής (AppKey) στον διακομιστή εφαρμογής.
- Βήμα 5: Η συσκευή αποκρυπτογραφή το μήνυμα αποδοχής χρησιμοποιώντας το (AppKey - AppNonce) για να εξάγει τα δύο κλειδιά συνεδρίας (NwkSKey - AppSKey) τα οποία αποθηκεύονται στην συσκευή μαζί με την διεύθυνση που μας έδωσε ο διακομιστής (DevAddr), τώρα πλέον η συσκευή είναι ενεργοποιημένη



Σχήμα 36: Διάγραμμα ροής OTAA v1.0

12 Συμπεράσματα και Παρατηρήσεις

Μέχρι την ίδρυση και ανάπτυξη του πρωτοκόλλου LoRa WAN τα δίκτυα IoT που μπορούσαμε να διαμορφώσουμε ήταν είτε κλειστά δηλαδή δεν είχαν καμία επικοινωνία με κάποιο άλλο δίκτυο παρά μόνο εσωτερικά ή ήταν συνδεδεμένα με μεθόδους που είχαν μειονεκτήματα ή περιπλοκότητα ή έλλειψη ασφάλειας π.χ (χρήση τηλεφωνικού δικτύου κυψέλης, WiFi).

Η ίδρυση ενός δικτύου αφιερωμένο απόλυτα για την διασύνδεση IoT υποδικτύων και συσκευών μεταξή τους αλλά και με το internet δημιούργησε μια πληθώρα επιλογών και λύσεων σε σχεδόν όλους τους τομείς της ζωής μας.
Δίνοντας μας έλεγχο και παρακολούθηση σε απομακρυσμένα σημεία και εγκαταστάσεις αυξάνοντας έτσι τις δυνατότητες μας.
Μερικοί από αυτούς τους τομείς είναι:

- Παρακολούθηση της αλυσίδας εμβολίων: Οι αισθητήρες LoRaWAN μας επιτρέπουν την παρακολούθηση της θερμοκρασίας κατά την μεταφορά.
- Διατήρηση επαφής με ζώα προς εξαφάνιση: Γεωγραφική παρακολούθηση ζώων προς εξαφάνιση.
- Έξυπνες φάρμες: Παρακολούθηση σε πραγματικό χρόνο της υγρασίας του εδάφους βελτιώνοντας το πρόγραμμα ποτίσματος και κάνοντας εξοικονόμηση νερού έως και 30
- Εξοικονόμηση νερού: Άμεση ειδοποίηση σπατάλης νερού σε τυχόν βλάβες στους αγωγούς υδροδότησης μιας πόλης.
- Ασφάλεια τροφίμων: Παρακολούθηση και διατήρηση σταθερής θερμοκρασίας για την ποιότητα και ασφάλεια των τροφίμων.

12.1 Έγκριση από ITU

Κατά την εγγραφή αυτής της εργασίας (2022), πρόσφατα στης 7 Δεκεμβρίου 2021 το πρωτόκολλο LoRaWAN εγκρίθηκε από την παγκόσμια ένωση τηλεπικοινωνιών ITU(International Telecomunications Union) ως ένα στάνταρ για τα ευρείας περιοχής δίκτυα χαμηλής ισχύος, Low Power Wide Area Networking (LPWAN).

12.2 Μετρήσεις και βελτιώσεις

Κατά τον σχεδιασμό και κατασκευή του σταθμού, επιλέχθηκαν οι μέθοδοι επικοινωνίας και ο εξοπλισμός σύμφωνα με οικονομικά κριτήρια και κριτήρια διαθεσιμότητας τόσο για τον εξοπλισμό όσο και για την απουσία του δικτύου LoRaWAN.

12.3 Βελτιώσεις

Αυτό δεν σημαίνει όμως πως δεν υπάρχει χώρος για βελτίωση του σταθμού.
Μια επιλογή που μπορεί να βελτιώσει τον σταθμό είναι να προσθέσουμε και άλλα αισθητήρια όργανα όπως:

- Ανεμοδείκτης
- Αισθητήριο ταχύτητας ανέμου
- Αισθητήριο βροχόπτωσης
- Αισθητήριο ηλιακής ακτινοβολίας

Πολλά και διάφορα ακόμα τα οποία βελτιώνουν τον σταθμό αλλά βεβαίως αυξάνουν το κόστος και την πολυπλοκότητα του.

12.4 Μετρήσεις

Αυτή την στιγμή ο σταθμός τροφοδοτείται από ένα τροφοδοτικό $5VDC1.2A$, και η κατανάλωση ρεύματος του έρχεται στα $2.5mA$ με αποτέλεσμα η ισχύς που απαιτεί να είναι $1.25W$.
Μια πρόταση θα ήταν να δημιουργήσουμε μία αυτονομία για κάποιες μέρες σε περίπτωση που δεν είναι διαθέσιμη η ηλεκτροδότηση του σταθμού.

Με τα δεδομένα που έχουμε μπορούμε να βρούμε πώς η κατανάλωση του σταθμού ανέρχεται στις:

Ημερησίως
$1.25W \cdot 24h = 30Wh/day$
Εβδομαδιαία
$30Wh \cdot 7d = 210Wh/week$
Μηνιαία
$30Wh \cdot 30d = 900Wh/month$

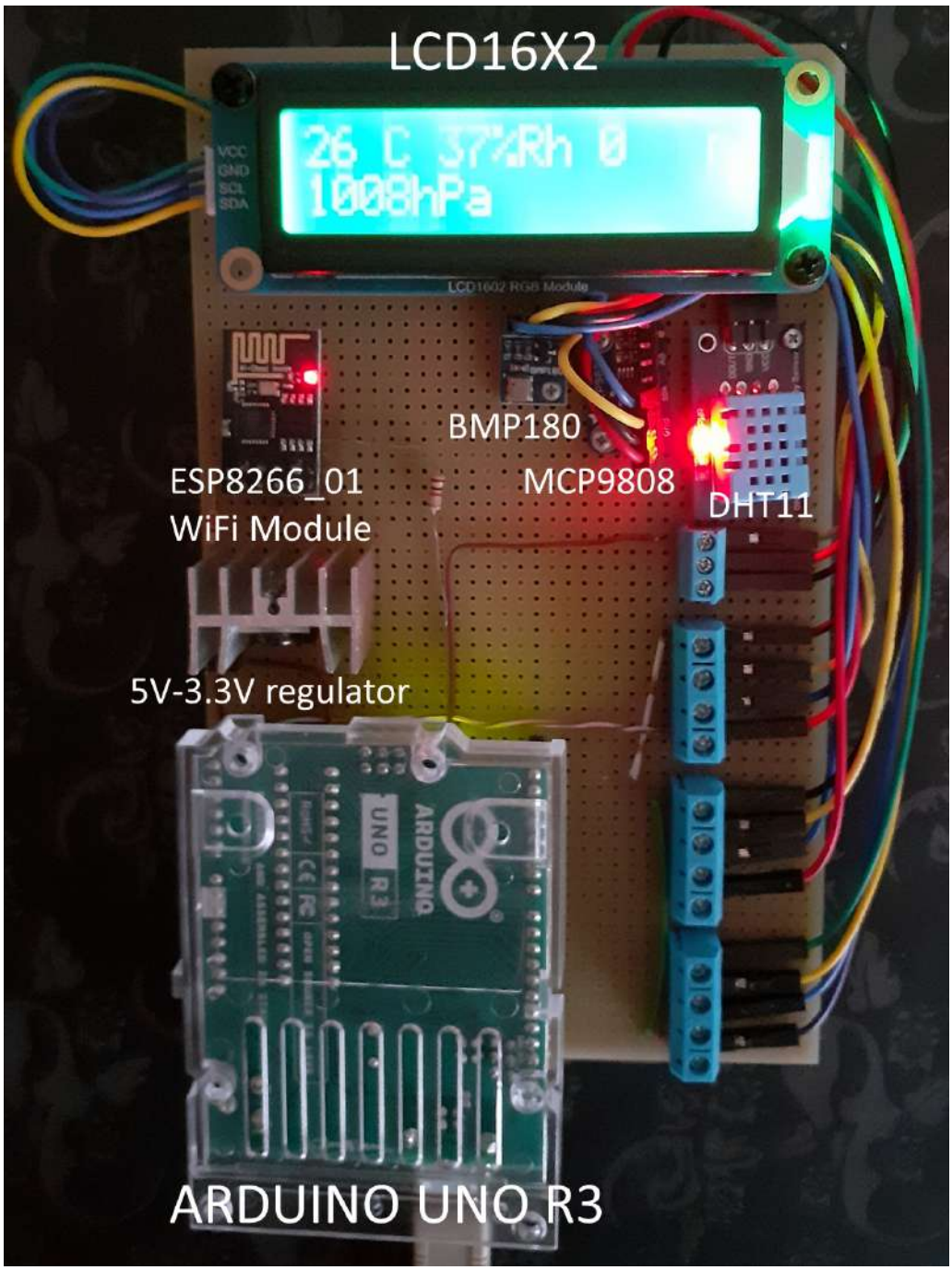
Αναλόγως με την περιοχή που θα τοποθετούσαμε τον σταθμό θα έχουμε και το ποσοστό ηλιοφάνειας της περιοχής αλλά μια γενική περίπτωση 7 ημέρες απουσία ηλιοφάνειας στους πιο χειμερινούς μήνες είναι αρκετή.

Έτσι θα χρειαζόμασταν μπαταρίες:
$210Wh/5V = 42Ah$

Καθώς και ένα Φ/Β πάνελ όσο η διπλάσια ισχύς του σταθμού με μία αύξηση 30% λόγο απωλειών:
$(2,5W \cdot 0.3) + 2,5W = 3,25W$

Φυσικά θα επιλέξουμε σύμφωνα με τις επιλογές του εμπορίου, η πλησιέστερη επιλογή που έχουμε είναι Φ/Β πάνελ των $5W$.

13 Φωτογραφία Σταθμού



14 Βιβλιογραφία

References

[1] LoRa Alliance. *What is LoRaWAN Specification*. URL: https://lora-alliance.org/about-lorawan/. (accessed: 08.2016).

[2] WiFi Alliance. *WiFi Alliance*. URL: https://www.wi-fi.org/who-we-are. (accessed: 09.2016).

[3] *Analog Barometer*. URL: https://res.cloudinary.com/dtpgi0zck/image/upload/s--VaxzO8s1--/c_fit,h_580,w_860/v1/EducationHub/photos/barometer.jpg.

[4] *Analog Thermometer*. URL: https://www.freepik.com/premium-photo/analogue-wooden-thermometer-with-red-measuring-liquid-black-background_26977068.htm.

[5] Arduino. *Arduino UNO R3*. URL: https://docs.arduino.cc/hardware/uno-rev3?queryID=undefined.

[6] *Bandwidth to range graph*. URL: https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan/bandwidth-vs-range.png.

[7] Bosch. *Barometric pressure sensors*. URL: https://www.bosch-sensortec.com/products/environmental-sensors/pressure-sensors/. (accessed: 07.2016).

[8] IEE. *IEEE 802 Standards*. URL: https://ieeexplore.ieee.org/browse/standards/get-program/page/series?id=68. (accessed: 09.2016).

[9] The Things Network. *End Device Activation*. URL: https://www.thethingsnetwork.org/docs/lorawan/end-device-activation/. (accessed: 09.2022).

[10] Patrick Neis Herman G. J. Smit Susanne Rohs Ulrich Bundke Martina Krämer Nicole Spelten Volker Ebert Bernhard Buchholz Karin Thomas Andreas Petzold. *Quality assessment of MOZAIC and IAGOS capacitive hygrometers: insights from airborne field studies*. URL: https://doi.org/10.3402/tellusb.v67.28320. (accessed: 07.2022).

[11] *Pressure Sensor*. URL: https://www.bosch-sensortec.com/media/boschsensortec/products/productrenderings_16_9/16_21/barometic_pressure_sensors/bosch-sensortec-products-bmp280-16-9_res_1280x720.jpg.

[12] N.D. Gohar S.A. Moiz Kh.S. Karimov. *Orange Dye Thin Film Resistive Hygrometers*. URL: https://www.ect-journal.kz/index.php/ectj/article/view/594. (accessed: 07.2022).

[13] *TTN network architecture*. URL: https://www.thethingsnetwork.org/docs/lorawan/architecture/architecture.png.

[14] Klaus Witrisal. *OFDM for Wireless Communications Systems*. International series of monographs on physics. Artech House, 2004. ISBN: 1-58053-796-0.

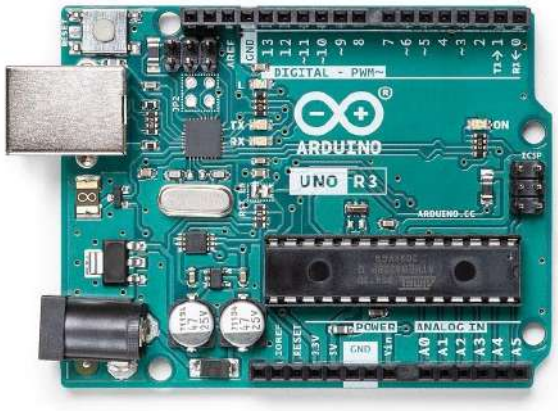
[15] Hossam Labib Zayed et al. «Effect of switching loads on the power line communication(PLC) modems inside direct local area network (dlan)». In: Dec. 2014.

Κατάλογος Σχημάτων

1 αναλογικό θερμόμετρο οινοπνεύματος 7
2 ψηφιακό θερμόμετρο 7
3 Αναλογικό Βαρόμετρο 9
4 ψηφιακός αισθητήρας πίεσης 9
5 γράφημα σχέσης εύρους φάσματος - συχνότητας 14
6 Αρχιτεκτονική του δυκτίου LoRaWAN 15
7 α) απλή πολυπλεξία β) ορθογώνια πολυπλεξία 17
8 Function Block Diagram 19
9 κύκλωμα σταθμού 20
10 Αρχιτεκτονική δικτύου ThingSpeak 61
11 ThingSpeak Βήμα 1ο 61
12 ThingSpeak Βήμα 1.1 62
13 ThingSpeak Βήμα 2ο 62
14 ThingSpeak Βήμα 3ο 63
15 ThingSpeak Βήμα 3.1 64
16 ThingSpeak Βήμα 3.2 65
17 ThingSpeak Βήμα 4ο 65
18 ThingSpeak Βήμα 4.1 66
19 ThingSpeak Βήμα 4.2 67
20 ThingSpeak Βήμα 5ο 68
21 ThingSpeak Όργανα ένδειξης 68
22 ThingSpeak Γραφήματα 69
23 ThingSpeak Παράδειγμα οργάνων 70
24 ThingSpeak Παράδειγμα γραφημάτων 70
25 Δίκτυο TTN 71
26 TTN Βήμα 1ο 72
27 TTN Βήμα 1.1 72
28 TTN Βήμα 2ο 73
29 TTN Βήμα 2.1 73
30 TTN Βήμα 2.2 74
31 TTN Βήμα 2.3 75
32 TTN Βήμα 2.4 75
33 TTN Βήμα 3ο 76
34 TTN Βήμα 3.1 77
35 Βήμα 3.2 78
36 Διάγραμμα ροής OTAA v1.0 80

15 Παράρτημα

Product Reference Manual
SKU: A000066



Description

The Arduino UNO R3 is the perfect board to get familiar with electronics and coding. This versatile microcontroller is equipped with the well-known ATmega328P and the ATMega 16U2 Processor.
This board will give you a great first experience within the world of Arduino.

Target areas:

Maker, introduction, industries

BMP180

DIGITAL PRESSURE SENSOR

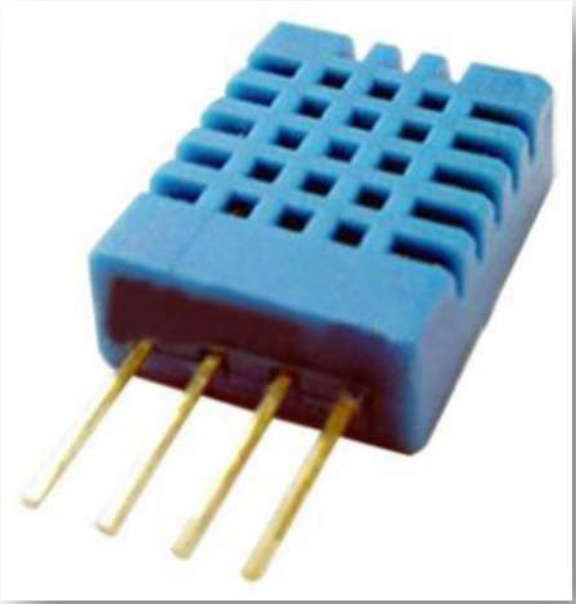
Key features

Pressure range:	300 ... 1100hPa (+9000m ... -500m relating to sea level)
Supply voltage:	1.8 ... 3.6V (V_{DD}) 1.62V ... 3.6V (V_{DDIO})
Package:	LGA package with metal lid Small footprint: 3.6mm x 3.8mm Super-flat: 0.93mm height
Low power:	5µA at 1 sample / sec. in standard mode
Low noise:	0.06hPa (0.5m) in ultra low power mode 0.02hPa (0.17m) advanced resolution mode

- Temperature measurement included
- I^2C interface
- Fully calibrated
- Pb-free, halogen-free and RoHS compliant,
- MSL 1

Typical applications

- Enhancement of GPS navigation (dead-reckoning, slope detection, etc.)
- In- and out-door navigation
- Leisure and sports
- Weather forecast
- Vertical velocity indication (rise/sink speed)



Each DHT11 element is strictly calibrated in the laboratory that is extremely accurate on humidity calibration. The calibration coefficients are stored as programmes in the OTP memory, which are used by the sensor's internal signal detecting process. The single-wire serial interface makes system integration quick and easy. Its small size, low power consumption and up-to-20 meter signal transmission making it the best choice for various applications, including those most demanding ones. The component is 4-pin single row pin package. It is convenient to connect and special packages can be provided according to users' request.

2. Technical Specifications:

Overview:

Item	Measurement Range	Humidity Accuracy	Temperature Accuracy	Resolution	Package
DHT11	20-90%RH 0-50 ℃	±5%RH	±2℃	1	4 Pin Single Row

1. Overview

Espressif's ESP8266EX delivers highly integrated Wi-Fi SoC solution to meet users' continuous demands for efficient power usage, compact design and reliable performance in the Internet of Things industry.

With the complete and self-contained Wi-Fi networking capabilities, ESP8266EX can perform either as a standalone application or as the slave to a host MCU. When ESP8266EX hosts the application, it promptly boots up from the flash. The integrated high-speed cache helps to increase the system performance and optimize the system memory. Also, ESP8266EX can be applied to any microcontroller design as a Wi-Fi adaptor through SPI/SDIO or UART interfaces.

ESP8266EX integrates antenna switches, RF balun, power amplifier, low noise receive amplifier, filters and power management modules. The compact design minimizes the PCB size and requires minimal external circuitries.

Besides the Wi-Fi functionalities, ESP8266EX also integrates an enhanced version of Tensilica's L106 Diamond series 32-bit processor and on-chip SRAM. It can be interfaced with external sensors and other devices through the GPIOs. Software Development Kit (SDK) provides sample codes for various applications.

Espressif Systems' Smart Connectivity Platform (ESCP) enables sophisticated features including:

- Fast switch between sleep and wakeup mode for energy-efficient purpose;
- Adaptive radio biasing for low-power operation
- Advance signal processing
- Spur cancellation and RF co-existence mechanisms for common cellular, Bluetooth, DDR, LVDS, LCD interference mitigation

1.1. Wi-Fi Key Features

- 802.11 b/g/n support
- 802.11 n support (2.4 GHz), up to 72.2 Mbps
- Defragmentation
- 2 x virtual Wi-Fi interface
- Automatic beacon monitoring (hardware TSF)
- Support Infrastructure BSS Station mode/SoftAP mode/Promiscuous mode

MCP9808

±0.5°C Maximum Accuracy Digital Temperature Sensor

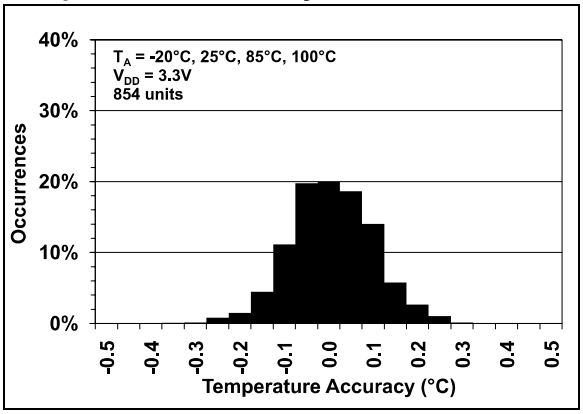
Features

- Accuracy:
  - ±0.25 (typical) from -40°C to +125°C
  - ±0.5°C (maximum) from -20°C to 100°C
  - ±1°C (maximum) from -40°C to +125°C
- User-Selectable Measurement Resolution:
  - +0.5°C, +0.25°C, +0.125°C, +0.0625°C
- User-Programmable Temperature Limits:
  - Temperature Window Limit
  - Critical Temperature Limit
- User-Programmable Temperature Alert Output
- Operating Voltage Range: 2.7V to 5.5V
- Operating Current: 200 µA (typical)
- Shutdown Current: 0.1 µA (typical)
- 2-wire Interface: I²C™/SMBus Compatible
- Available Packages: 2x3 DFN-8, MSOP-8

Typical Applications

- General Purpose
- Industrial Applications
- Industrial Freezers and Refrigerators
- Food Processing
- Personal Computers and Servers
- PC Peripherals
- Consumer Electronics
- Handheld/Portable Devices

Temperature Accuracy



Description

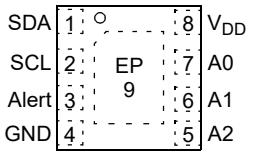
Microchip Technology Inc.'s MCP9808 digital temperature sensor converts temperatures between -20°C and +100°C to a digital word with ±0.25°C/±0.5°C (typical/maximum) accuracy.

The MCP9808 comes with user-programmable registers that provide flexibility for temperature sensing applications. The registers allow user-selectable settings such as Shutdown or Low-Power modes and the specification of temperature Alert window limits and critical output limits. When the temperature changes beyond the specified boundary limits, the MCP9808 outputs an Alert signal. The user has the option of setting the Alert output signal polarity as an active-low or active-high comparator output for thermostat operation, or as a temperature Alert interrupt output for microprocessor-based systems. The Alert output can also be configured as a critical temperature output only.

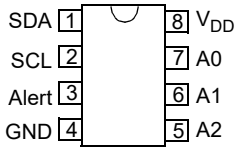
This sensor has an industry standard 400 kHz, 2-wire, SMBus/I²C compatible serial interface, allowing up to eight or sixteen sensors to be controlled with a single serial bus (see Table 3-2 for available Address codes). These features make the MCP9808 ideal for sophisticated, multi-zone, temperature-monitoring applications.

Package Types

8-Pin 2x3 DFN*

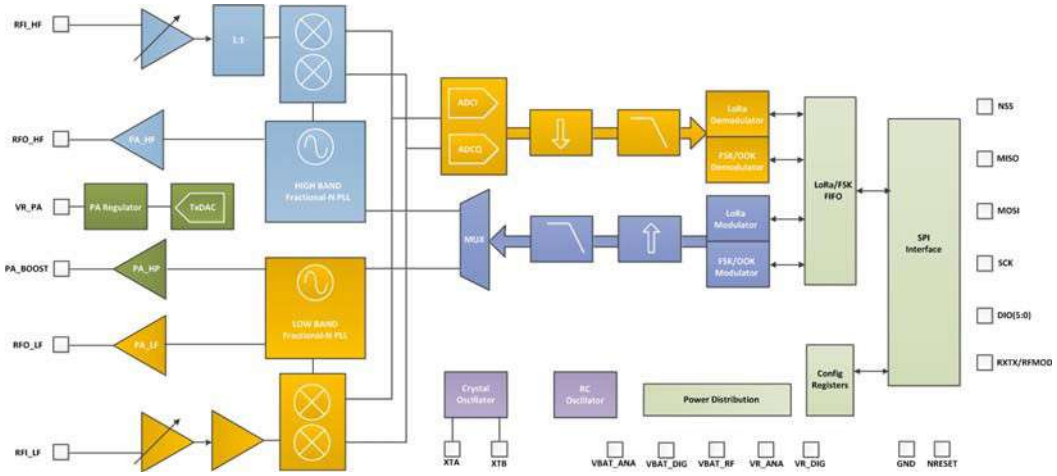


8-Pin MSOP



* Includes Exposed Thermal Pad (EP); see Table 3-1.

SX1276/77/78/79 - 137 MHz to 1020 MHz Low Power Long Range Transceiver



GENERAL DESCRIPTION

The SX1276/77/78/79 transceivers feature the LoRa™ long range modem that provides ultra-long range spread spectrum communication and high interference immunity whilst minimising current consumption.

Using Semtech's patented LoRa™ modulation technique SX1276/77/78/79 can achieve a sensitivity of over -148dBm using a low cost crystal and bill of materials. The high sensitivity combined with the integrated +20 dBm power amplifier yields industry leading link budget making it optimal for any application requiring range or robustness. LoRa™ provides significant advantages in both blocking and selectivity over conventional modulation techniques, solving the traditional design compromise between range, interference immunity and energy consumption.

These devices also support high performance (G)FSK modes for systems including WMBus, IEEE802.15.4g. The SX1276/77/78/79 deliver exceptional phase noise, selectivity, receiver linearity and IIP3 for significantly lower current consumption than competing devices.

ORDERING INFORMATION

Part Number	Delivery	MOQ / Multiple
SX1276IMLTRT	T&R	3000 pieces
SX1277IMLTRT	T&R	3000 pieces
SX1278IMLTRT	T&R	3000 pieces
SX1279IMLTRT	T&R	3000 pieces
SX1276WS	Wafer Form	1 Wafer (2000 dies)

- QFN 28 Package - Operating Range [-40;+85°C]
- Pb-free, Halogen free, RoHS/WEEE compliant product

KEY PRODUCT FEATURES

- LoRa™ Modem
- 168 dB maximum link budget
- +20 dBm - 100 mW constant RF output vs. V supply
- +14 dBm high efficiency PA
- Programmable bit rate up to 300 kbps
- High sensitivity: down to -148 dBm
- Bullet-proof front end: IIP3 = -11 dBm
- Excellent blocking immunity
- Low RX current of 9.9 mA, 200 nA register retention
- Fully integrated synthesizer with a resolution of 61 Hz
- FSK, GFSK, MSK, GMSK, LoRa™and OOK modulation
- Built-in bit synchronizer for clock recovery
- Preamble detection
- 127 dB Dynamic Range RSSI
- Automatic RF Sense and CAD with ultra-fast AFC
- Packet engine up to 256 bytes with CRC
- Built-in temperature sensor and low battery indicator

APPLICATIONS

- Automated Meter Reading.
- Home and Building Automation.
- Wireless Alarm and Security Systems.
- Industrial Monitoring and Control
- Long range Irrigation Systems