ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Εφαρμογή διασύνδεσης ενός μικρού μετεωρολογικού σταθμού με κινητό τηλέφωνο Android

Κυριακούλα Α. Μπουρλίνου

Εισηγητής: Ευθύμιος Αλέπης, Καθηγητής

(Κενό φύλλο)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Εφαρμογή διασύνδεσης ενός μικρού μετεωρολογικού σταθμού με κινητό τηλέφωνο Android

Κυριακούλα Α. Μπουρλίνου
Α.Μ. ais0083

Εισηγητής:

Δρ Ευθύμιος Αλέπης, Καθηγητής

Εξεταστική Επιτροπή:

Ιωάννης Έλληνας, Καθηγητής

Δρ Ευθύμιος Αλέπης, Καθηγητής

Ημερομηνία εξέτασης 10/07/2019

(Κενό φύλλο)

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Η κάτωθι υπογεγραμμένη Μπουρλίνου Κυριακούλα, του Αλεξάνδρου, με αριθμό μητρώου **ais0083** φοιτήτρια του Τμήματος Μηχανικών Η/Υ Συστημάτων Τ.Ε. του Α.Ε.Ι. Πειραιά Τ.Τ. πριν αναλάβω την εκπόνηση της Πτυχιακής Εργασίας μου, δηλώνω ότι ενημερώθηκα για τα παρακάτω:

«Η Πτυχιακή Εργασία (Π.Ε.) αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο του συγγραφέα, όσο και του Ιδρύματος και θα πρέπει να έχει μοναδικό χαρακτήρα και πρωτότυπο περιεχόμενο.

Απαγορεύεται αυστηρά οποιοδήποτε κομμάτι κειμένου της να εμφανίζεται αυτούσιο ή μεταφρασμένο από κάποια άλλη δημοσιευμένη πηγή. Κάθε τέτοια πράξη αποτελεί προϊόν λογοκλοπής και εγείρει θέμα Ηθικής Τάξης για τα πνευματικά δικαιώματα του άλλου συγγραφέα. Αποκλειστικός υπεύθυνος είναι ο συγγραφέας της Π.Ε., ο οποίος φέρει και την ευθύνη των συνεπειών, ποινικών και άλλων, αυτής της πράξης.

Πέραν των όποιων ποινικών ευθυνών του συγγραφέα σε περίπτωση που το Ίδρυμα του έχει απονείμει Πτυχίο, αυτό ανακαλείται με απόφαση της Συνέλευσης του Τμήματος. Η Συνέλευση του Τμήματος με νέα απόφασης της, μετά από αίτηση του ενδιαφερόμενου, του αναθέτει εκ νέου την εκπόνηση της Π.Ε. με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή. Η εκπόνηση της εν λόγω Π.Ε. πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός ημερολογιακού 6μήνου από την ημερομηνία ανάθεσης της. Κατά τα λοιπά εφαρμόζονται τα προβλεπόμενα στο άρθρο 18, παρ. 5 του ισχύοντος Εσωτερικού Κανονισμού.»

(Κενό φύλλο)

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή εργασία ολοκληρώθηκε μετά από διεξοδική έρευνα σε δύο καινοτόμα και ενδιαφέροντα τεχνολογικά πεδία. Το ένα πεδίο αφορά τον τομέα της μικροηλεκτρονικής και τις καινούργιες πλατφόρμες που συνεχώς εξελίσσονται, με έμφαση στις πλατφόρμες του Arduino και τις δυνατότητές τους για διασύνδεση με διάφορα εξαρτήματα. Το άλλο πεδίο αφορά τον εξίσου εξελισσόμενο τομέα του προγραμματισμού και συγκεκριμένα της προγραμματιστικής γλώσσας java μέσα από την προσαρμοσμένη έκδοσή της για οικιακές συσκευές, Android. Την προσπάθειά μου αυτή υποστήριξε ο επιβλέπων καθηγητής μου, τον οποίο θα ήθελα να ευχαριστήσω, για τον χρόνο του και την υπομονή του.

(Κενό φύλλο)

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία ασχολείται με την δημιουργία μιας οικιακής συσκευής που θα μετρά και θα καταγράφει την θερμοκρασία και την υγρασία του χώρου για δύο διαφορετικές τοποθεσίες. Η συσκευή αυτή θα επικοινωνεί ηλεκτρονικά με μια ιστοσελίδα στην οποία θα αποθηκεύει τις πληροφορίες αυτές. Επίσης θα αναπτυχθεί μια εφαρμογή για χρήση μέσω κινητού τηλεφώνου, με την οποία θα καθίσταται δυνατή η παρακολούθηση της πορείας των μετρήσεων με την βοήθεια γραφημάτων.

ABSTRACT

This diploma thesis deals with the creation of a home device that measures and records the temperature and humidity of the space for two different locations. This device will be able to communicate electronically with a webpage, where it will store all the necessary information. Also, a mobile phone application will be developed, which will allow the user to track the measurements and monitor the temperature and humidity changes through visualized graphics.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Αρχιτεκτονική Ηλεκτρονικών Υπολογιστών
ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: arduino, nodeMCU, android, μετεωρολογικός σταθμός

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

1.1 Περιγραφή του αντικειμένου της πτυχιακής εργασίας

Σκοπός της συγκεκριμένης εργασίας είναι να παρουσιάσει έναν οικιακό μετεωρολογικό σταθμό, με αισθητήρες μέτρησης σε δύο χώρους (εσωτερικό και εξωτερικό χώρο) και την αλληλεπίδρασή τους με μια Android εφαρμογή, με στόχο την παρατήρηση αλλά και την διαχείριση των καιρικών φαινομένων.

Η παρούσα εργασία ασχολείται με δύο τεχνολογικά πεδία της σημερινής εποχής. Το πρώτο μέρος αναφέρεται στη διαδικασία προγραμματισμού της υπολογιστικής πλατφόρμας Arduino και της υπολογιστικής πλατφόρμας Node MCU, με σκοπό την αξιοποίηση της ως αυτόνομο οικιακό μετεωρολογικό σταθμό, με δυνατότητες καταγραφής, επεξεργασίας και αποθήκευσης των μετεωρολογικών δεδομένων της περιοχής, με σκοπό την παρακολούθηση του καιρού σύμφωνα με τα δεδομένα που έχουν συλλεχθεί.

Αρχικά γίνεται μια ανάλυση της κάθε πλατφόρμας και στην συνέχεια παρουσιάζονται οι αισθητήρες που χρησιμοποιήθηκαν, τα κριτήρια επιλογής τους, ο τρόπος συνδεσμολογίας και οι βιβλιοθήκες που αξιοποιήθηκαν για την επικοινωνία τους με τις δύο προαναφερόμενες πλακέτες.

Γίνεται επεξήγηση της διαδικασίας καταγραφής με την βοήθεια της ανοιχτής πλατφόρμας ThingSpeak από την εταιρία Matlab και αναλύεται ο τρόπος με τον οποίο ο μετεωρολογικός σταθμός επικοινωνεί με την πλατφόρμα.

Για να διαπιστωθεί η αξιοπιστία των αισθητήρων καθώς και των αλγορίθμων καταγραφής, γίνεται σύγκριση των δεδομένων που συλλέχθηκαν με τα αντίστοιχα από υπάρχον πιστοποιημένο μετεωρολογικό σταθμό της περιοχής, το αποτέλεσμα της οποίας είναι ικανοποιητικό.

Το δεύτερο μέρος της παρούσας εργασίας ασχολείται με την παρουσίαση των συλλεγόμενων στοιχείων σε μια εφαρμογή Android και τους δυνατούς τρόπους αξιοποίησης τους. Δίνεται βάση στην επεξήγηση της εφαρμογής, στον σχεδιασμό της και στην έμμεση αλληλεπίδρασή της με τον μετεωρολογικό σταθμό.

ΚΕΦΑΛΑΙΟ 2

ΠΛΑΤΦΟΡΜΑ ARDUINO

2.1 Περιγραφή της πλατφόρμας Arduino

Η πλατφόρμα Arduino είναι μια ηλεκτρονική πλατφόρμα ανοιχτού κώδικα και βασίζεται σε μια λογική εύκολης χρήσης υλικού και λογισμικού. Οι πλατφόρμες Arduino είναι σε θέση να διαβάσουν εισόδους π.χ. φως σε έναν αισθητήρα ή ένα δάχτυλο σε ένα κουμπί και να το μετατρέψουν σε ένα αξιοποιήσιμο σήμα εξόδου π.χ. για την ενεργοποίηση ενός μοτέρ ή ενός LED. Για τον προγραμματισμό τους χρησιμοποιείται το λογισμικό Arduino IDE και η γλώσσα προγραμματισμού Wiring.

Με τα χρόνια το Arduino έχει καταφέρει να γίνει η βάση για χιλιάδες καθημερινά project, αλλά και για σύνθετα επιστημονικά πειράματα. Αυτό έχει επιτευχθεί λόγω του ότι η συγκεκριμένη πλατφόρμα είναι «ανοιχτού κώδικα», κάτι το οποίο σημαίνει πως άνθρωποι όλων των ειδικοτήτων μπορούν να δημιουργήσουν κάτι και να το μοιραστούν με άλλους.

Το Arduino δημιουργήθηκε στο Ivrea Interaction Design Institute σαν ένα εύκολο εργαλείο για ταχεία πρωτοτυποποίηση, στοχεύοντας μαθητές με υπόβαθρο στα ηλεκτρονικά και στον προγραμματισμό. Όταν έγινε ευρέως γνωστό άρχισε να προσαρμόζεται στις ανάγκες του κοινού του, κάνοντάς το σήμερα μια διαδεδομένη ηλεκτρονική πλατφόρμα κατάλληλη για ένα μεγάλο εύρος ανθρώπων και έργων.

2.2 Πλεονεκτήματα της πλατφόρμας Arduino

Παρακάτω είναι μερικά από τα πλεονεκτήματα της πλατφόρμας Arduino που την βοήθησαν στην εξέλιξή της.

- Οικονομική – Οι πλατφόρμες Arduino είναι πολύ πιο οικονομικές συγκριτικά με άλλες πλατφόρμες μικροελεγκτών. Οι πιο απλοί ξεκινούν από ποσά κάτω των 5€ και μπορούν να φτάσουν τα 100€ με πολλές ενδιάμεσες επιλογές.
- Πολλαπλής πλατφόρμας – Το λογισμικό που χρησιμοποιείται (Arduino Software (IDE)) είναι συμβατό με Windows, Macintosh OSX, και Linux λειτουργικό

σύστημα. Τα περισσότερα λογισμικά για μικροελεγκτές περιορίζονται στα Windows.

- Απλό προγραμματιστικό περιβάλλον – Το Arduino Software (IDE) είναι εύκολο στην χρήση για κάποιον αρχάριο, χωρίς όμως να περιορίζει και έναν πιο έμπειρο χρήστη. Ακολουθεί την δομή της ευρέως διαδεδομένης C++ κάνοντάς το ιδανικό ακόμα και για μαθητές.
- Πρόγραμμα ανοιχτού κώδικα – Το Arduino Software (IDE) είναι ένα πρόγραμμα ανοιχτού κώδικα, διαθέσιμο σε έμπειρους προγραμματιστές. Έχοντας σαν δεδομένο αυτό και το ότι επεκτείνεται με βιβλιοθήκες της C++, καταλήγουμε να έχουμε ένα λογισμικό με πολύ μεγάλες δυνατότητες.
- Επεκτάσιμα εξαρτήματα – Όλα τα σχέδια για τις πλατφόρμες Arduino είναι διαθέσιμα στο κοινό, κάτω από άδειες Creative Commons, οπότε ο κάθε ένας μπορεί να χρησιμοποιήσει όποια έκδοση θέλει, να την επεκτείνει και να την βελτιστοποιήσει.

2.3 Εκδόσεις της πλατφόρμας Arduino

Στον Πίνακα 2.1 εμφανίζονται κάποιες από τις βασικές εκδόσεις του Arduino με μερικά από τα κύρια χαρακτηριστικά τους.

Name	Processor	Operating Voltage	CPU Speed	Analog In/ Out	Digital IO/ PWM	Flash (KB)
Uno	ATmega328	5 V/7-12 V	16MHz	6/0	14/6	32
Due	AT91SAM3X8E	3.3 V/7-12 V	84 MHz	12/2	54/12	512
Leonardo	ATmega32u4	5 V/7-12 V	16MHz	12/0	20/7	32
Mega 2560	ATmega2560	5 V/7-12 V	16MHz	16/0	54/15	256
Mega ADK	ATmega2560	5 V/7-12 V	16MHz	16/0	54/15	256
Micro	ATmega32u4	5 V/7-12 V	16MHz	12/0	20/7	32
Mini	ATmega328	5 V/7-9 V	16MHz	8/0	14/6	32
Nano	ATmega168/ATmega328	5 V/7-9 V	16MHz	8/0	14/6	16/32
Ethernet	ATmega328	5 V/7-12 V	16MHz	6/0	14/4	32
Esplora	ATmega32u4	5 V/7-12 V	16MHz	-	-	32
ArduinoBT	ATmega328	5 V/2.5-12 V	16MHz	6/0	14/6	32
Fio	ATmega328P	3.3 V/3.7-7 V	8MHz	8/0	14/6	32
Pro (168)	ATmega168	3.3 V/3.35-12 V	8MHz	6/0	14/6	16
Pro (328)	ATmega328	5 V/5-12 V	16MHz	6/0	14/6	32
Pro Mini	ATmega168	3.3 V/3.35-12 V 5 V/5-12 V	8MHz 16MHz	6/0	14/6	16
LilyPad	ATmega168V/ATmega328V	2.7-5.5 V/2.7-5.5 V	8MHz	6/0	14/6	16
LilyPad USB	ATmega32u4	3.3 V/3.8-5V	8MHz	4/0	9/4	32
LilyPad Simple	ATmega328	2.7-5.5 V/2.7-5.5 V	8MHz	4/0	9/4	32
LilyPad SimpleSnap	ATmega328	2.7-5.5 V/2.7-5.5 V	8MHz	4/0	9/4	32
Yun	ATmega32u4	5 V	16MHz	12/0	20/7	32

Πίνακας 2.1 – Εκδόσεις Arduino

2.4 Η πλατφόρμα Arduino Nano

Στα πλαίσια της παρούσας εργασίας χρησιμοποιήθηκε η έκδοση Arduino Nano. Η έκδοση αυτή έχει τα ακόλουθα χαρακτηριστικά.

Microcontroller	ATmega328
Architecture	AVR
Operating Voltage	5 V
Flash Memory	32 KB of which 2 KB used by bootloader
SRAM	2 KB
Clock Speed	16 MHz
Analog IN Pins	8
EEPROM	1 KB
DC Current per I/O Pins	40 mA (I/O Pins)
Input Voltage	7-12 V
Digital I/O Pins	22 (6 of which are PWM)
PWM Output	6
Power Consumption	19 mA
PCB Size	18 x 45 mm
Weight	7 g

Πίνακας 2.2 – Χαρακτηριστικά Arduino Nano

Το Arduino Nano αποτελεί την πιο μικρή πλακέτα της τεχνολογίας Arduino και είναι η μικρή εκδοχή του Arduino Uno. Είναι πολύ διαδεδομένο μοντέλο και συμβατό με πλήθος αισθητήρων και επεκτάσεων.

Η λειτουργία του είναι ακριβώς ίδια με την λειτουργία του Arduino Uno σε μικρότερο μέγεθος και η πλακέτα είναι πλήρως συμβατή ώστε να τοποθετηθεί σε πλακέτες δοκιμών (breadboard).

Το Arduino Nano βασίζεται στον μικροελεγκτή ATmega328 της Atmel. Είναι μια ολοκληρωμένη πλακέτα που περιέχει ότι χρειάζεται για να μπορεί να προγραμματιστεί και να λειτουργήσει συνδέοντάς την με ένα απλό καλώδιο Mini-B USB στον υπολογιστή.

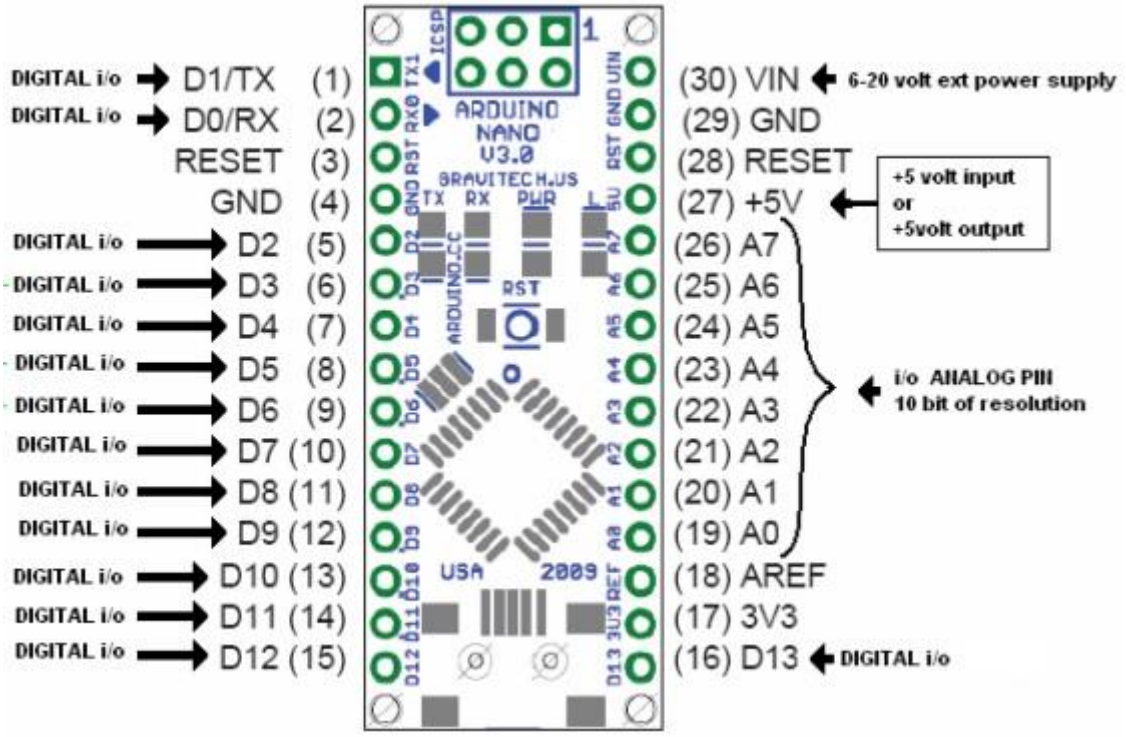
Αναλυτικά η πλακέτα διαθέτει 14 ψηφιακές εισόδους ή εξόδους (6 απο αυτές μπορεί να χρησιμοποιηθούν σαν PWM - Pulse Width Modulation - εξόδους), 6 αναλογικές εισόδους, 1 θύρα Mini-B USB για τον προγραμματισμό και την τροφοδοσία της πλακέτας, 1 υποδοχή ICSP - In-Circuit Serial Programming - και

τέλος κουμπί για το reset της πλακέτας. Ο μικροελεγκτής είναι συγχρονισμένος στους 16 μεγάκυκλους (Crystal 16MHz).

Η μνήμη Flash όπου μπορεί να αποθηκεύσει ένα πρόγραμμα (sketch) είναι 32KB, ικανή να δεχτεί τα περισσότερα απλά προγράμματα.

Τέλος το Arduino Nano λειτουργεί σε χαμηλή τάση, με τροφοδοσία 5V DC από την είσοδο του USB, χωρίς να υπάρχει κίνδυνος ηλεκτροπληξίας.

Στην Εικόνα 2.1 παρουσιάζονται οι είσοδοι και οι έξοδοι του Arduino Nano.



Εικόνα 2.1 - Είσοδοι/Έξοδοι Arduino Nano

Η συγκεκριμένη πλακέτα χρησιμοποιήθηκε για την μέτρηση των μετεωρολογικών φαινομένων στον εξωτερικό χώρο, γιατί είναι πολύ μικρή σε μέγεθος και καταναλώνει λιγότερο ρεύμα με αποτέλεσμα να έχει μεγαλύτερη αυτονομία.

ΚΕΦΑΛΑΙΟ 3

ΠΛΑΤΦΟΡΜΑ NODEMCU-ESP8266

3.1 Περιγραφή του μικροελεγκτή ESP8266

Η πλατφόρμα NodeMCU (Node MicroController Unit) περιέχει ένα λογισμικό ανοιχτού κώδικα και ένα περιβάλλον ανάπτυξης υλικού, που είναι αναπτυγμένα σύμφωνα με τον μικροελεγκτή ESP8266. Ο μικροελεγκτής ESP8266 έχει σχεδιαστεί και υλοποιηθεί από την Espressif Systems και περιέχει όλα τα κύρια στοιχεία ενός σύγχρονου υπολογιστή: CPU, RAM, δυνατότητες δικτύου (wifi) και ένα ανεξάρτητο σύστημα ανάπτυξης λογισμικού (Software Development Kit – SDK). Η τιμή του σήμερα κυμαίνεται περίπου στα 5€, κάτι που τον κάνει πολύ οικονομικό και συμφέρων για όλων των ειδών τα project.

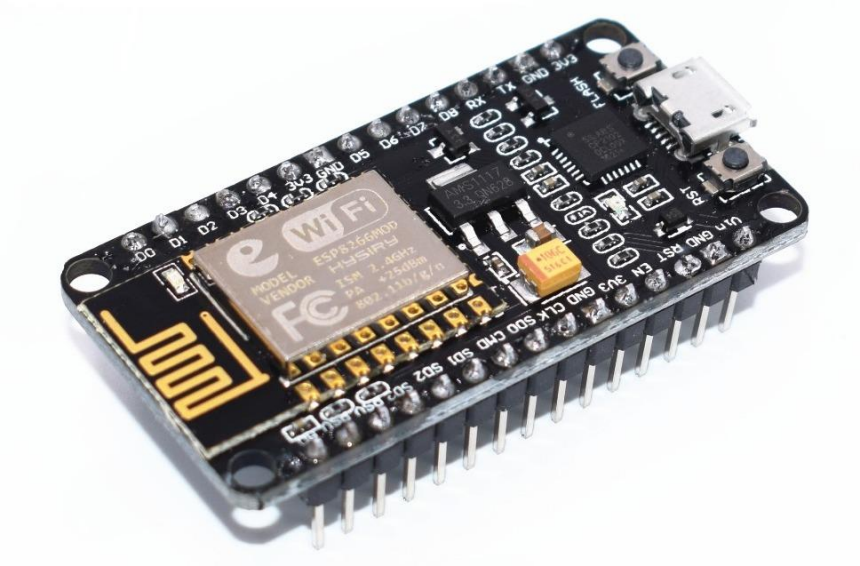
Παρόλα αυτά, σαν μικροελεγκτής, ο ESP8266 είναι επίσης δύσκολος στην πρόσβαση και στην διαχείριση. Ακόμα και για να τεθεί σε λειτουργία θα πρέπει κάποιος να ενώσει καλώδια με κατάλληλη αναλογική τάση σε όλα του τα PINs και για να το προγραμματίσει θα πρέπει να χρησιμοποιήσει γλώσσα μηχανής χαμηλού επιπέδου.

3.2 Πλεονεκτήματα της πλατφόρμας NodeMCU

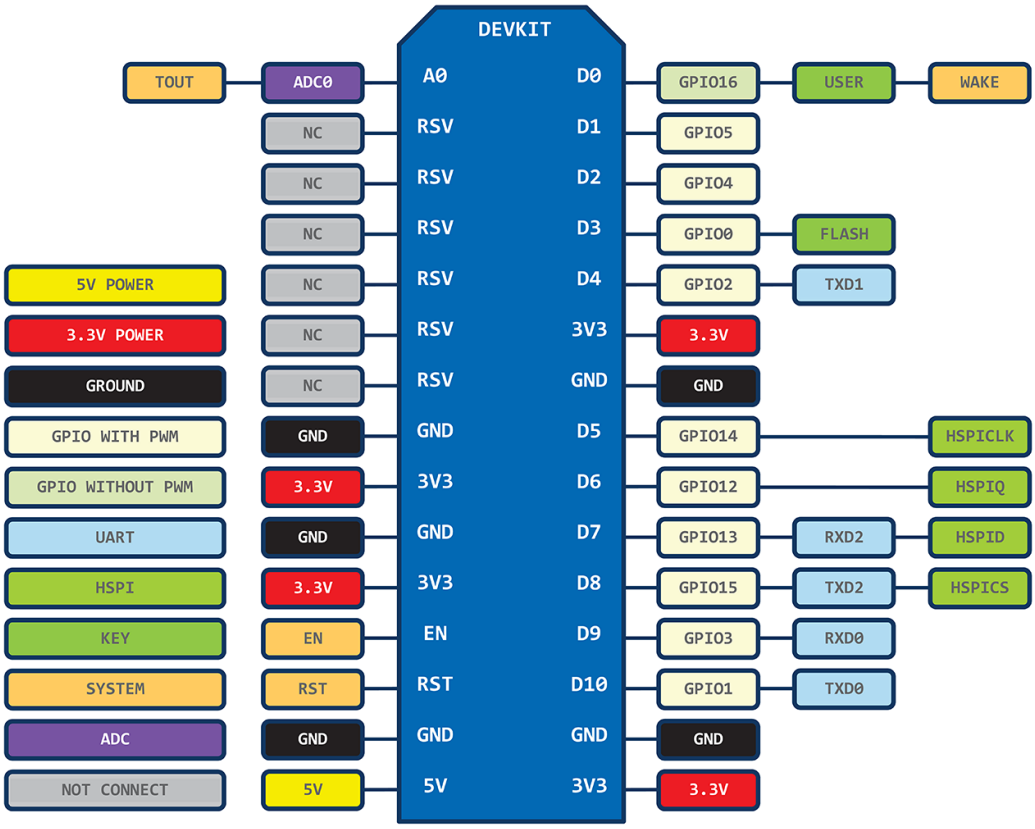
Τα παραπάνω εμπόδια έρχεται να λύσει η πλατφόρμα NodeMCU. Η συγκεκριμένη πλατφόρμα έχει δύο σημαντικά πλεονεκτήματα:

- Ένα ανοιχτού λογισμικού firmware που είναι προσαρμοσμένο πάνω στο SDK του κατασκευαστή και προσφέρει ένα απλό προγραμματιστικό περιβάλλον. Ο προγραμματισμός του μπορεί να γίνει επίσης από το περιβάλλον του Arduino IDE, δίνοντας την δυνατότητα να προγραμματίζεται ακόμα και από αρχάριους.
- Μια πλατφόρμα στην οποία είναι ήδη ενσωματωμένο το chip, έχοντας καλωδιωμένα σωστά όλα του τα PINs. Έχει επίσης συνδεδεμένη μια θύρα USB, ένα κουμπί επανεκκίνησης, κεραία wifi, ενδεικτική λυχνία λειτουργίας και επεκτάσεις PINs που μπορούν να τοποθετηθούν σε πειραματικές διατάξεις(bread board).

Η Εικόνα 3.1 απεικονίζει το NodeMCU ESP8266 και η Εικόνα 3.2 απεικονίζει τις εισόδους και τις εξόδους του.



Εικόνα 3.1 - NodeMCU ESP8266 module



Εικόνα 3.2 - NodeMCU ESP8266 Pin out

ΚΕΦΑΛΑΙΟ 4

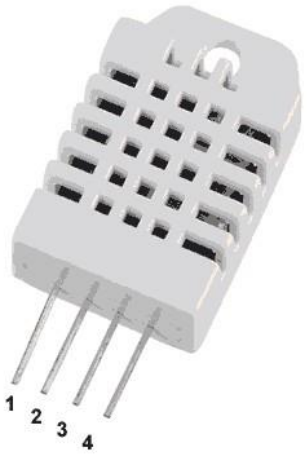
ΑΙΣΘΗΤΗΡΑΣ DHT22

4.1 Περιγραφή του αισθητήρα DHT22

Ο αισθητήρας DHT22 είναι ένας χαμηλού κόστους αισθητήρας που μετράει την υγρασία και την θερμοκρασία του χώρου. Χρησιμοποιεί έναν αισθητήρα υγρασίας και έναν αισθητήρα θερμοκρασίας για να μετράει τον περιβάλλοντα χώρο και παράγει ένα ψηφιακό σήμα στο pin δεδομένων χωρίς να χρειάζεται κάποια αναλογική είσοδο. Είναι πολύ εύκολος στην χρήση, αλλά χρειάζεται προσοχή στην χρονική στιγμή άντλησης δεδομένων. Αυτό είναι και το μόνο του μειονέκτημα γιατί μπορεί να στέλνει καινούργια δεδομένα κάθε 2 δευτερόλεπτα χωρίς ο χρόνος αυτός να μπορεί να μειωθεί περαιτέρω.

Η Εικόνα 4.1 παρουσιάζει έναν αισθητήρα DHT22 και τους ακροδέκτες του.

DHT22 pins	
1	VCC
2	DATA
3	NC
4	GND



Εικόνα 4.1 – Αισθητήρας DHT22

4.2 Συνδεσμολογία του αισθητήρα DHT22

Η συνδεσμολογία του είναι απλή. Το πρώτο pin από αριστερά, τοποθετείται στην τροφοδοσία, το δεύτερο pin από αριστερά είναι η έξοδος των ψηφιακών δεδομένων και το πρώτο από δεξιά τοποθετείται στην γείωση. Ανάμεσα στην τροφοδοσία και το πρώτο Pin θα πρέπει να τοποθετηθεί και μία αντίσταση 4.7K - 10K για την προστασία του αισθητήρα.

4.3 Πλεονεκτήματα του αισθητήρα DHT22

- Υψηλή ακρίβεια.
- Χωρητικού τύπου.
- Πλήρες αντισταθμισμένο φάσμα θερμοκρασίας.
- Μέτρηση σχετικής υγρασίας και θερμοκρασίας.
- Βαθμονομημένο ψηφιακό σήμα.
- Μακροπρόθεσμη σταθερότητα.
- Χωρίς ανάγκη επιπλέον στοιχείων.
- Μεγάλη απόσταση μετάδοσης (μέχρι 100m).
- Χαμηλή κατανάλωση ρεύματος.

4.4 Τεχνικά χαρακτηριστικά του αισθητήρα DHT22

Μοντέλο:	AM2302
Παροχή ενέργειας:	3.3-5.5V DC
Σήμα εξόδου:	digital signal via 1-wire bus
Στοιχείο ανίχνευσης:	Polymer humidity capacitor
Εύρος λειτουργίας:	humidity 0-100%RH; temperature -40~80Celsius
Ακρίβεια:	humidity +-2%RH(Max +-5%RH); temperature +-0.5Celsius
Ανάλυση/Ευαισθησία:	humidity 0.1%RH; temperature 0.1Celsius
Επαναληψημότητα:	humidity +-1%RH; temperature +-0.2Celsius
Υστέρηση υγρασίας:	+-0.3%RH
Μακροπρόθεσμη σταθερότητα:	+-0.5%RH/year
Ανταλλαξιμότητα:	fully interchangeable

Πίνακας 4.1 – Τεχνικά Χαρακτηριστικά του DHT22

ΚΕΦΑΛΑΙΟ 5

ΡΟΛΟΙ ΠΡΑΓΜΑΤΙΚΟΥ ΧΡΟΝΟΥ (RTC) DS3231

5.1 Περιγραφή του RTC DS3231

Το DS3231 είναι ένα χαμηλού κόστους και υπερβολικά ακριβές I2C Real Time Clock (RTC), με έναν ολοκληρωμένο αντισταθμισμένο ταλαντωτή κρυστάλλου (Temperature Compensated Crystal Oscillator-TCXO). Ο TCXO ταλαντωτής παρέχει ένα σταθερό και ακριβές ρολόι αναφοράς ρολόι και διατηρεί το RTC να είναι εντός των ± 2 λεπτών ακρίβειας ανά έτος για θερμοκρασίες από-40 ° C έως + 85 ° C.

Η συσκευή έχει μια ενσωματωμένη μια είσοδο μπαταρίας η οποία εξασφαλίζει την διατήρηση της ακριβής χρονομέτρησης σε περίπτωση διακοπής της τροφοδοσίας της κύριας συσκευής. Το RTC διατηρεί πληροφορία για τα δευτερόλεπτα, τα λεπτά, τις ώρες, τις ημέρες, τις ημερομηνίες, τους μήνες και τους χρόνους. Η ημερομηνία προσαρμόζεται αυτόματα στους μήνες με ημέρες λιγότερες των 31, όπως επίσης προσαρμόζεται ανάλογα και με το αν το έτος είναι δίσεκτο.

Το ρολόι λειτουργεί είτε σε 12ωρη είτε σε 24ωρη λειτουργία έχοντας διαθέσιμη και την ένδειξη AM/PM. Παρέχονται δύο προγραμματιζόμενα alarms για στιγμές μέσα στην ημέρα και μία προγραμματιζόμενη έξοδο τετραγωνικού παλμού. Διευθύνσεις και δεδομένα μεταφέρονται σειριακά μέσω ενός I2C δίαυλου διπλής κατεύθυνσης.

Στην Εικόνα 5.1 παρουσιάζεται ένα ρολόι πραγματικού χρόνου DS3231.

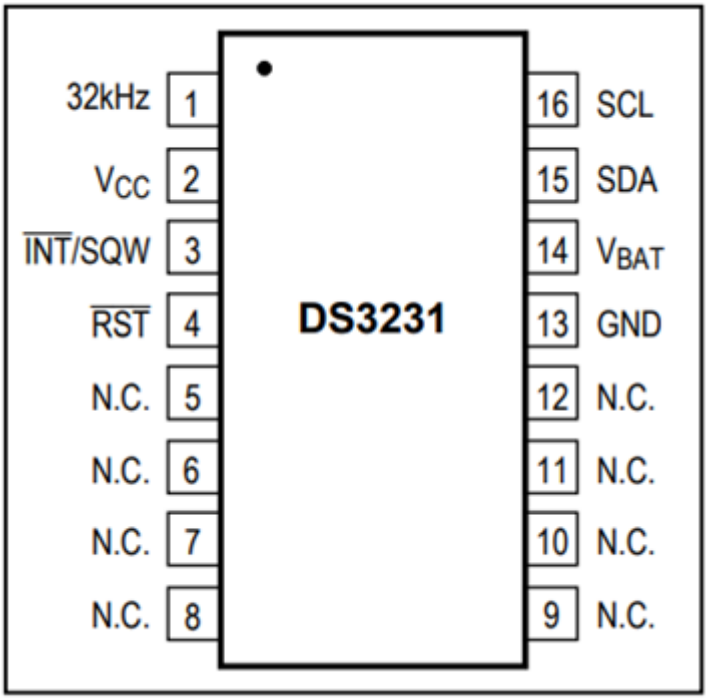
Εικόνα 5.1 – Ρολόι Πραγματικού Χρόνου DS3231

5.2 Πλεονεκτήματα και χαρακτηριστικά του RTC DS3231

- ✓ Ιδιαίτερα ακριβές RTC που διαχειρίζεται πλήρως όλες τις λειτουργίες χρονομέτρησης με προσαρμοσμένες ημερομηνίες μέχρι το 2100.
- ✓ Ακρίβεια ±2ppm από 0°C μέχρι +40°C.
- ✓ Ακρίβεια ±3.5ppm από -40°C μέχρι +85°C.
- ✓ Ψηφιακή έξοδος αισθητήρα θερμοκρασίας: ±3°C ακρίβεια.
- ✓ RST Έξοδος/Κουμπί Εισόδου επαναφοράς.
- ✓ Δύο alarm μέσα στην ίδια ημέρα.
- ✓ Προγραμματιζόμενο σήμα εξόδου τετραγωνικού παλμού.
- ✓ Απλή σειριακή διασύνδεση με τους περισσότερους μικροελεκτές.
- ✓ Γρήγορη διασύνδεση I2C (400kHz).
- ✓ Εφεδρική μπαταρία για την συνοχή του χρόνου.
- ✓ Λειτουργία χαμηλής κατανάλωσης ρεύματος - επεκτείνει το χρόνο λειτουργίας της εφεδρικής μπαταρίας.
- ✓ Λειτουργία 3.3V.
- ✓ Εύρος θερμοκρασίας λειτουργίας: Εμπορική (0°C to +70°C), Βιομηχανική (-40°C to +85°C).
- ✓ Αναγνωρισμένο από τα εργαστήρια Underwriters Laboratories® (UL).

5.3 Περιγραφή των ακροδεκτών του RTC DS3231

Η Εικόνα 5.2 εμφανίζει μια οπτική παρουσίαση της διάταξης των ακροδεκτών του ολοκληρωμένου DS3231και ο Πίνακας 5.1 δίνει μία πιο αναλυτική περιγραφή της χρήσης τους.



Εικόνα 5.2 – DS3231 Pin Out

Αριθμός PIN	*Όνομα*	*Περιγραφή*
1	32kHz	Έξοδος 32kHz. Αυτό το pin χρειάζεται εξωτερική pullup αντίσταση. Όταν ενεργοποιηθεί, η έξοδος λειτουργεί σαν τροφοδοτικό. Μπορεί να παραμείνει ασύνδετο αν δεν χρησιμοποιείται.
2	VCC	Pin DC ρεύματος για κύρια τροφοδοσία. Αυτό το pin θα πρέπει να αποσυνδεθεί χρησιμοποιώντας πυκνωτή 0.1μF με 1.0μF. Αν δεν χρησιμοποιηθεί, θα πρέπει να γειωθεί.
3	INT/SQW	Active-Low Interrupt (Διακοπή χαμηλής ενεργοποίησης) ή Square-Wave Output (έξοδος τετραγωνικού παλμού). Αυτό το pin χρειάζεται εξωτερική pullup αντίσταση συνδεδεμένη με τροφοδοσία 5.5V ή λιγότερο. Αυτό το πολλαπλών λειτουργιών pin καθορίζει την λειτουργία του από το Bit INTCN στο μητρώο ελέγχου (0Eh). Όταν το INTCN είναι ρυθμισμένο στο λογικό 0, το Pin λειτουργεί σαν έξοδος τετραγωνικού παλμού, ενώ όταν είναι ρυθμισμένο στο λογικό 1, λειτουργεί σαν διακοπή χαμηλής ενεργοποίησης.

4	RST	Επαναφορά χαμηλής ενεργοποίησης. Αυτό το pin μπορεί να ενεργοποιηθεί από ένα κουμπί επαναφοράς. Έχει μία εσωτερική pullup αντίσταση με ονομαστική τιμή 50kΩ πριν την τροφοδοσία. Δεν χρειάζεται εξωτερική pullup αντίσταση.
5-12	N.C.	Χωρίς χρήση. Πρέπει να συνδέεται με την γείωση.
13	GND	Γείωση.
14	VBAT	Είσοδος εφεδρικής τροφοδοσίας. Όταν η συσκευή χρησιμοποιεί το VBAT σαν κύρια πηγή τροφοδοσίας, το pin θα πρέπει να συνδεθεί με έναν πυκνωτή 0.1μF έως 1.0μF. Όταν χρησιμοποιείται το VBAT σαν είσοδος εφεδρικής τροφοδοσίας, ο πυκνωτής δεν είναι απαραίτητος.
15	SDA	Σειριακά δεδομένα Εισόδου/Εξόδου. Αυτό το Pin είναι η είσοδος/έξοδος για τα δεδομένα της σειριακής διασύνδεσης I2C. Αυτό το pin χρειάζεται εξωτερική pullup αντίσταση. Η pullup τάση μπορεί να είναι μέχρι 5.5V, ανεξάρτητα από την τάση VCC.
16	SCL	Είσοδος σειριακού ρολογιού. Αυτό το pin είναι το ρολόι για την σειριακή διεπαφή I2C και χρησιμοποιείται για να συγχρονίζει τις κινήσεις των σειριακών δεδομένων. Η τάση μπορεί να είναι μέχρι 5.5V, ανεξάρτητα από την τάση VCC.

Πίνακας 5.1 – DS3231 Pin Out

5.4 Χρήση του RTC DS3231

Το module DS3231 έχει διάφορες χρήσεις, όμως στην παρούσα πτυχιακή εργασία αξιοποιήθηκαν οι δυνατότητές του ως real time clock. Το RTC παρέχει στοιχεία από δευτερόλεπτα έως και έτος. Προσαρμόζει επίσης την ημερομηνία ακόμα και σύμφωνα με το δίσεκτο έτος.

Οι πληροφορίες της ώρας και του ημερολογίου αποκτούνται με την ανάγνωση των κατάλληλων byte μητρώου. Τα δεδομένα του χρόνου και του ημερολογίου ορίζονται γράφοντας στα κατάλληλα bytes. Τα περιεχόμενα των καταχωρητών της ώρα και του ημερολογίου είναι σε μορφή BCD (δεκαδικός δυαδικού κώδικα). Το DS3231 μπορεί να τρέξει σε 12-ωρη ή 24-ωρη λειτουργία. Αυτό καθορίζεται από το Bit 6. Στη 12-ωρη λειτουργία, το bit 5 καθορίζει το AM/PM. Στην 24-ωρη λειτουργία, το bit 5 είναι το bit που περιέχει τις τιμές για τις ώρες 20 έως 23. Το bit που καθορίζει τον αιώνα (το bit 7 από τον καταχωρητή του μήνα) ενεργοποιείται όταν ο καταχωρητής που χρησιμοποιείται για τα χρόνια, ξεχειλίζει (πηγαίνει από το 99 στο 00). Ο μετρητής της ημέρας της εβδομάδας αλλάζει τα μεσάνυχτα. Οι τιμές που αντιστοιχούν στις ημέρες της εβδομάδας είναι οριζόμενες από το

χρήστη, αλλά πρέπει να είναι διαδοχικές (δηλ., εάν 1 ισούται με Κυριακή, τότε 2 ισούται με Δευτέρα, και ούτω καθεξής). Παράλογες τιμές ώρας και ημερομηνίας, θα έχουν ως αποτέλεσμα απροσδιόριστη λειτουργία.

Το DS3231 περιέχει δύο alarm για χρόνο μέσα στην ημέρα. Τα alarms μπορούν να προγραμματιστούν για να ενεργοποιήσουν την INT/SQW έξοδο, με το που ενεργοποιηθούν και αυτές. Οι καταχωρητές που χρησιμοποιούνται για τα alarms μπορούν να κάνουν δυνατούς πολλούς συνδυασμούς. Μπορούμε να έχουμε ειδοποιήσεις κάθε δευτερόλεπτο, κάθε 3 λεπτά, κάθε μιάμιση ώρα κ.λ.π. Η Εικόνα 5.3 δείχνει τους δυνατούς συνδυασμούς. Συνδυασμοί που δεν αναφέρονται στον πίνακα θα οδηγήσουν σε άγνωστη συμπεριφορά.

DY/$\overline{DT}$	ALARM 1 REGISTER MASK BITS (BIT 7)				ALARM RATE
	A1M4	A1M3	A1M2	A1M1	
X	1	1	1	1	Alarm once per second
X	1	1	1	0	Alarm when seconds match
X	1	1	0	0	Alarm when minutes and seconds match
X	1	0	0	0	Alarm when hours, minutes, and seconds match
0	0	0	0	0	Alarm when date, hours, minutes, and seconds match
1	0	0	0	0	Alarm when day, hours, minutes, and seconds match

DY/$\overline{DT}$	ALARM 2 REGISTER MASK BITS (BIT 7)			ALARM RATE
	A2M4	A2M3	A2M2	
X	1	1	1	Alarm once per minute (00 seconds of every minute)
X	1	1	0	Alarm when minutes match
X	1	0	0	Alarm when hours and minutes match
0	0	0	0	Alarm when date, hours, and minutes match
1	0	0	0	Alarm when day, hours, and minutes match

Εικόνα 5.3 – Δυνατοί Συνδυασμοί Alarm

ΚΕΦΑΛΑΙΟ 6

ΠΟΜΠΟΔΕΚΤΗΣ nRF24L01

6.1 Περιγραφή του nRF24L01

Το module nRF24L01 είναι ένας ραδιοφωνικός πομποδέκτης με ένα τσιπ για την παγκόσμια ζώνη ISM 2,4 - 2,5 GHz. Ο πομποδέκτης αποτελείται από ένα πλήρως ενσωματωμένο συνθεσάιζερ συχνοτήτων, έναν ενισχυτή ισχύος, έναν ταλαντωτή κρυστάλλου, έναν αποδιαμορφωτή, έναν διαμορφωτή και έναν πρωτόκολλο Enhanced ShockBurst ™ . Η ισχύς εξόδου, τα κανάλια συχνότητας και η ρύθμιση πρωτοκόλλου μπορούν εύκολα να προγραμματιστούν μέσω της διεπαφής SPI. Η τρέχουσα κατανάλωση είναι πολύ χαμηλή, μόνο 9.0mA με ισχύ εξόδου -6dBm και 12.3mA σε λειτουργία RX. Οι ενσωματωμένες λειτουργίες Power Down και Standby κάνουν την εξοικονόμηση ενέργειας εύκολα υλοποιήσιμη.

Εικόνα 6.1 – nRF24L01 Module1

6.2 Γενικά χαρακτηριστικά του nRF24L01

Η μονάδα πομποδέκτη nRF24L01 έχει σχεδιαστεί για να λειτουργεί σε ζώνη συχνότητας ISM παγκοσμίως 2,4 GHz και χρησιμοποιεί διαμόρφωση GFSK για τη μετάδοση δεδομένων. Ο ρυθμός μεταφοράς δεδομένων μπορεί να είναι ένας από τους 250kbps, 1Mbps και 2Mbps.

Η τάση λειτουργίας της μονάδας είναι από 1,9 έως 3,6V, αλλά οι ακίδες είναι ανθεκτικές στα 5 Volt, έτσι μπορούν να συνδεθούν εύκολα σε έναν μικροελεγκτή Arduino ή σε οποιοδήποτε λογικό μικροελεγκτή 5V χωρίς τη χρήση λογικού μετατροπέα τάσης εισόδου.

Η μονάδα πομποδέκτη nRF24L01 επικοινωνεί μέσω μιας σειριακής περιφερειακής διεπαφής (SPI) 4 ακίδων με μέγιστο ρυθμό δεδομένων 10Mbps. Όλες οι παράμετροι, όπως το κανάλι συχνότητας (125 επιλέξιμα κανάλια), μπορούν να διαμορφωθούν μέσω αυτής της διεπαφής. Ο δίαυλος διεπαφής SPI χρησιμοποιεί την νοοτροπία του Master/Slave, όπου στις κοινές εφαρμογές η συσκευή Arduino είναι η Master συσκευή και η μονάδα πομποδέκτη nRF24L01 είναι η Slave συσκευή.

Parameter	Value	Unit
Minimum supply voltage	1.9	V
Maximum output power	0	dBm
Maximum data rate	2000	kbps
Supply current in TX mode @ 0dBm output power	11.3	mA
Supply current in RX mode @ 2000 kbps	12.3	mA
Temperature range	-40 to +85	°C
Sensitivity @ 1000 kbps	-85	dBm
Supply current in Power Down mode	900	nA

Εικόνα 6.2 – Γενικά Χαρακτηριστικά του nRF24L01

6.3 Ακροδέκτες και τρόπος χρήσης του nRF24L01

Στην παρακάτω Εικόνα 6.3 εμφανίζονται οι ακροδέκτες του nRF24L01 και στον Πίνακα 6.1 υπάρχουν πληροφορίες για τον κάθε ακροδέκτη.

Εικόνα 6.3 – nRF24L01 Pin Out

Pin	Όνομα	Λειτουργία Pin	Περιγραφή
1	GND	Power	Γείωση
2	VCC	Power	Παροχή ενέργειας

3	CE	Digital Input	Chip Enable. Active-HIGH. Όταν επιλεγεί, το nRF24L01 θα μεταδίδει ή θα λαμβάνει, ανάλογα με τον τρόπο λειτουργίας που βρίσκεται αυτή τη στιγμή.
4	CSN	Digital Input	Chip Select Not. Active-LOW. Όταν χαμηλώνει, το nRF24L01 αρχίζει να ακούει στη θύρα SPI για δεδομένα και επεξεργάζεται ανάλογα.
5	SCK	Digital Input	Serial Clock. Δέχεται παλμούς ρολογιού που παρέχονται από το Master bus του SPI.
6	MOSI	Digital Input	Master Out Slave In. Η είσοδος SPI στο nRF24L01.
7	MISO	Digital Output	Master In Slave Out. Η έξοδος SPI από το nRF24L01.
8	IRQ	Digital Output	Ακροδέκτης διακοπής που ειδοποιεί τον master όταν υπάρχουν διαθέσιμα νέα δεδομένα για επεξεργασία.

Πίνακας 6.1 – nRF24L01 Pin Out

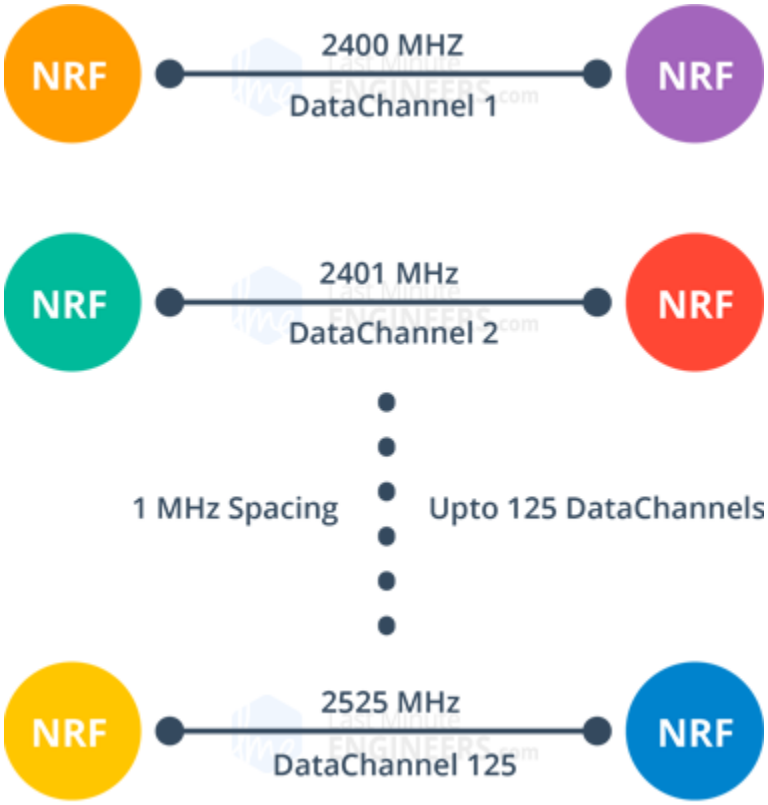
Η μονάδα πομποδέκτη nRF24L01 μεταδίδει και λαμβάνει δεδομένα σε μια συγκεκριμένη συχνότητα που ονομάζεται κανάλι. Επίσης, προκειμένου να επικοινωνούν μεταξύ τους δύο ή περισσότερες μονάδες πομποδέκτη, πρέπει να βρίσκονται στο ίδιο κανάλι. Αυτός ο δίαυλος θα μπορούσε να είναι οποιαδήποτε συχνότητα στη ζώνη ISM 2,4 GHz ή για την ακρίβεια, θα μπορούσε να είναι μεταξύ 2.400 και 2.525 GHz (2400 έως 2525 MHz).

Κάθε κανάλι καταλαμβάνει εύρος ζώνης μικρότερο από 1MHz. Αυτό μας δίνει 125 πιθανά κανάλια με απόσταση 1MHz. Έτσι, το module μπορεί να χρησιμοποιήσει 125 διαφορετικά κανάλια που δίνουν τη δυνατότητα να υπάρχει ένα δίκτυο 125 ανεξάρτητων λειτουργικών modems σε ένα μέρος.

Η συχνότητα καναλιού RF του επιλεγμένου καναλιού σας έχει οριστεί σύμφωνα με τον ακόλουθο τύπο:

$$Freq_{(Selected)} = 2400 + CH_{(Selected)}$$

Για παράδειγμα, εάν επιλεγεί το 108 ως το κανάλι για τη μετάδοση δεδομένων, η συχνότητα καναλιού RF θα είναι 2508MHz (2400 + 108).



Εικόνα 6.4 – Πιθανά κανάλια στο nRF24L01

ΚΕΦΑΛΑΙΟ 7

ΣΥΝΔΕΣΜΟΛΟΓΙΑ ΜΕΤΕΩΡΟΛΟΓΙΚΟΥ ΣΤΑΘΜΟΥ

7.1 Περιγραφή μετεωρολογικού σταθμού

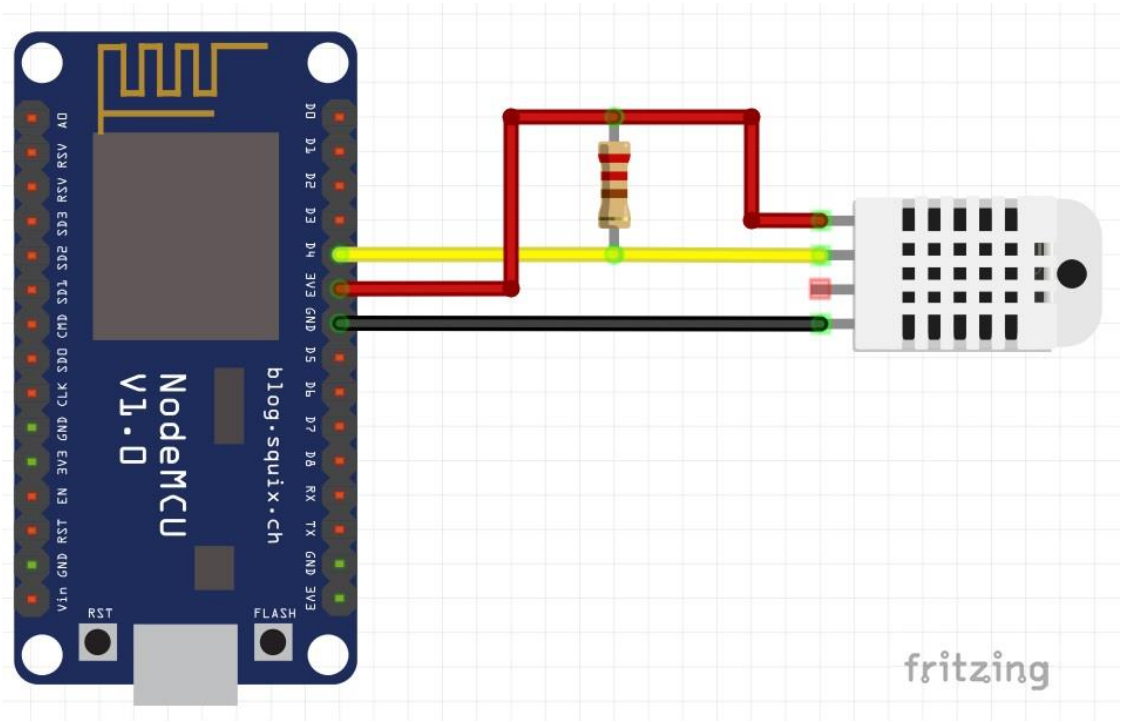
Για τις ανάγκες της παρούσας εργασίας, ο μετεωρολογικός σταθμός που δημιουργήθηκε αποτελείται από δύο τμήματα. Το ένα τμήμα είναι το τμήμα του πομπού (transmitter) και το άλλο είναι το τμήμα του δέκτη (receiver).

Το τμήμα του πομπού είναι αυτό που θα τοποθετηθεί σε απομακρυσμένο σημείο (π.χ. εκτός σπιτιού) και αποτελείται από μια πλατφόρμα Arduino nano, ένα ρολόι πραγματικού χρόνου RTC3231, έναν αισθητήρα θερμοκρασίας και υγρασίας DHT22 και έναν ραδιοφωνικό πομποδέκτη nRF24L01. Ο πομπός έχει προγραμματιστεί να καταγράφει ανά τακτά χρονικά διαστήματα την θερμοκρασία και την υγρασία του χώρου και μέσω του nRF24L01 να τα στέλνει στον δέκτη. Το ρολόι πραγματικού χρόνου χρησιμοποιείται για την καταγραφή της ώρας και της ημερομηνίας κάθε μέτρησης και για να ενεργοποιεί την πλατφόρμα Arduino nano ώστε να στέλνει τα δεδομένα στον δέκτη. Ο κώδικας που υλοποιήθηκε για τον πομπό βρίσκεται στο Παράρτημα Α’.

Το τμήμα του δέκτη είναι αυτό που θα τοποθετηθεί κοντά στο δίκτυο παροχής ιντερνέτ (π.χ. εντός σπιτιού) και αποτελείται από μια πλατφόρμα NodeMCU, ένα ρολόι πραγματικού χρόνου RTC3231, έναν αισθητήρα θερμοκρασίας και υγρασίας DHT22 και έναν ραδιοφωνικό πομποδέκτη nRF24L01. Ο δέκτης έχει προγραμματιστεί για να βρίσκεται στην αναμονή και να περιμένει να “ακούσει” τον πομπό. Όταν ο πομπός στείλει δεδομένα, ο δέκτης τα λαμβάνει μέσω του nRF24L01 και ταυτόχρονα μετράει την θερμοκρασία και την υγρασία του περιβάλλοντα χώρου με την βοήθεια του αισθητήρα. Το ρολόι πραγματικού χρόνου χρησιμοποιείται για την καταγραφή της ώρας και της ημερομηνίας κάθε μέτρησης. Η πλατφόρμα κάθε φορά που λαμβάνει ένα σετ μετρήσεων στέλνει ηλεκτρονικά ένα πακέτο δεδομένων και ενημερώνει την online πλατφόρμα ThingSpeak που αναλύεται παρακάτω. Ο κώδικας που υλοποιήθηκε για τον δέκτη βρίσκεται στο Παράρτημα Α’.

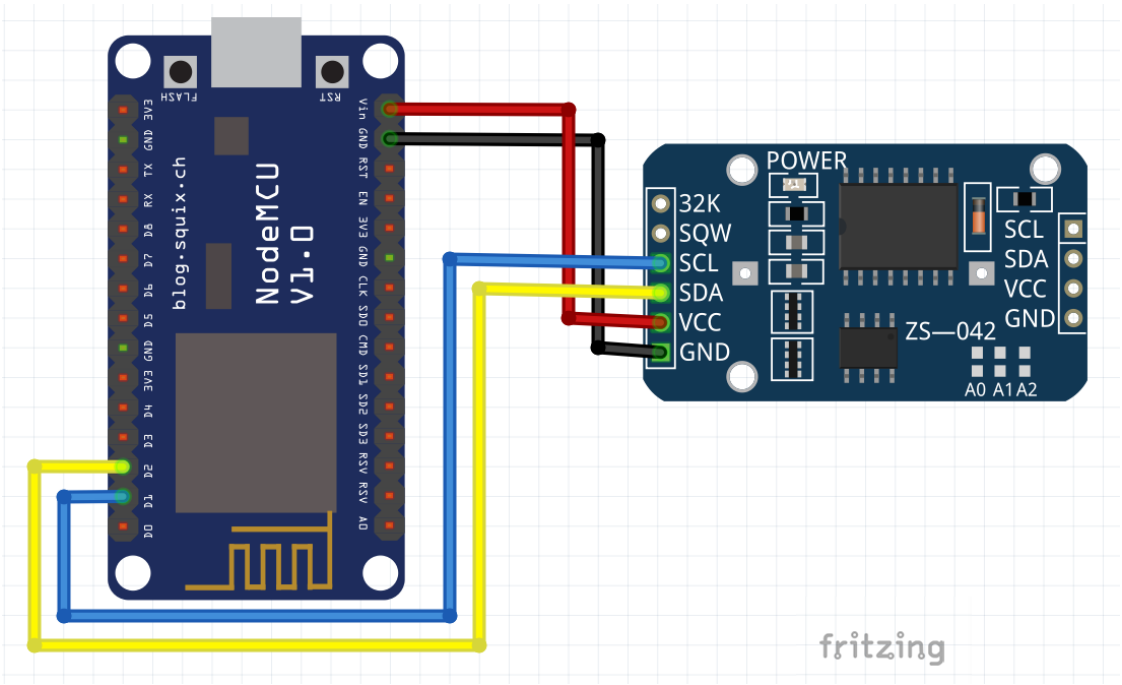
7.2 Συνδεσμολογία Receiver (Δέκτης)

Στην Εικόνα 7.1 εμφανίζεται η συνδεσμολογία της πλατφόρμας NodeMCU με τον αισθητήρα DHT22.



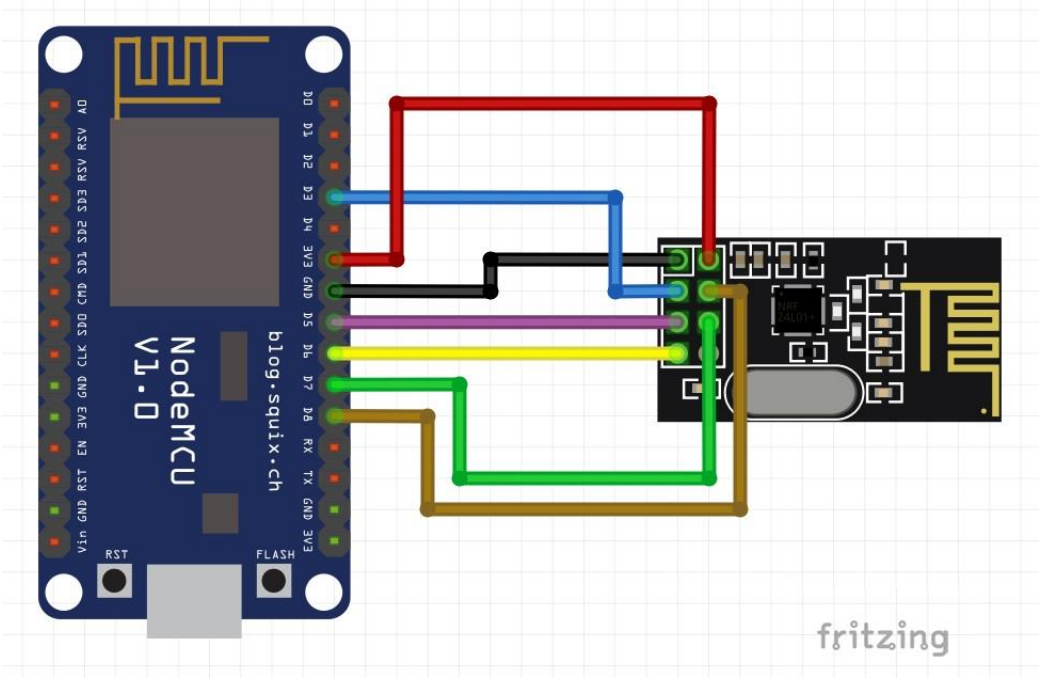
Εικόνα 7.1 – Συνδεσμολογία NodeMCU με DHT22

Στην Εικόνα 7.2 εμφανίζεται η συνδεσμολογία της πλατφόρμας NodeMCU με το ρολόι πραγματικού χρόνου RTC3231.



Εικόνα 7.2 – Συνδεσμολογία NodeMCU με RTC3231

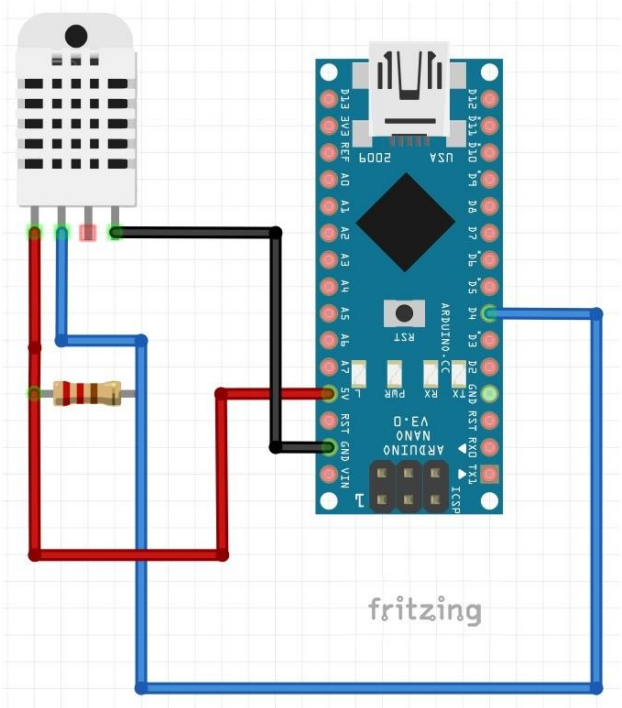
Στην Εικόνα 7.3 εμφανίζεται η συνδεσμολογία της πλατφόρμας NodeMCU με τον ραδιοφωνικό πομποδέκτη nRF24L01.



Εικόνα 7.3 – Συνδεσμολογία NodeMCU με nRF24L01

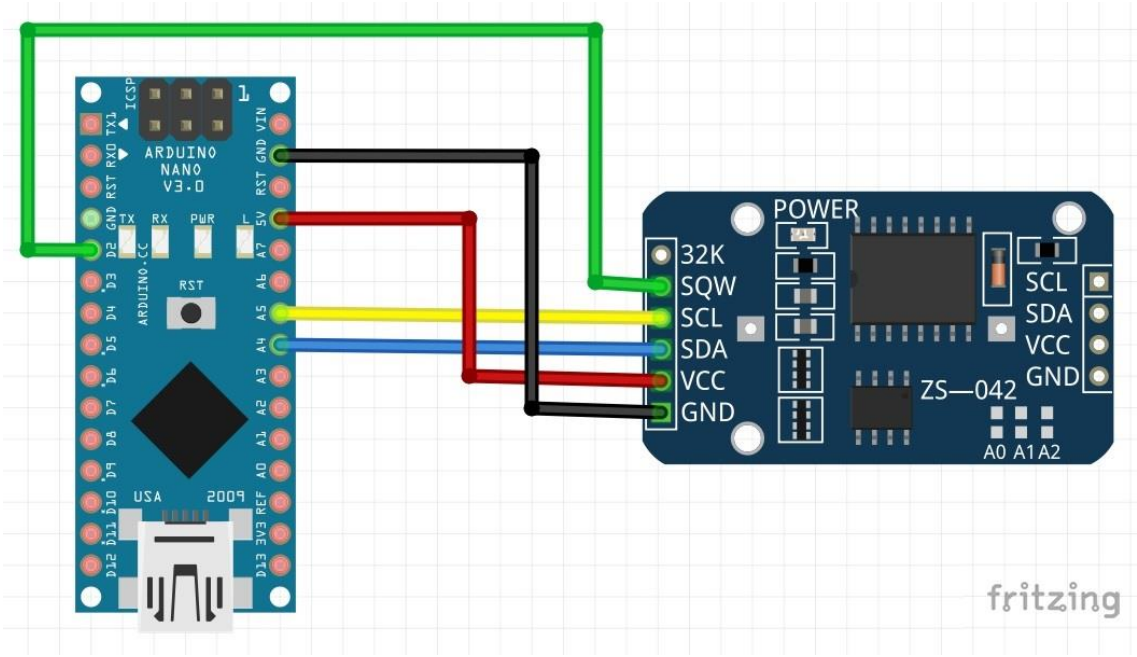
7.3 Συνδεσμολογία Transmitter (Πομπός)

Στην Εικόνα 7.4 εμφανίζεται η συνδεσμολογία της πλατφόρμας Arduino nano με τον αισθητήρα DHT22.



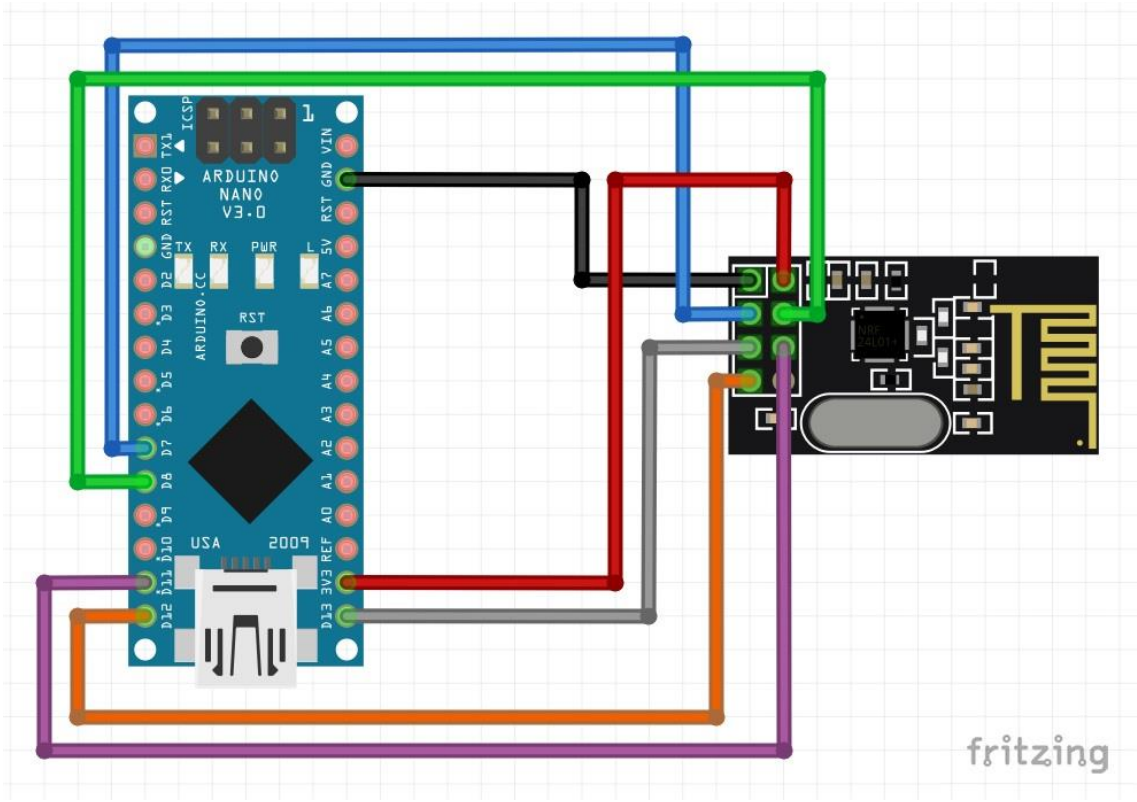
Εικόνα 7.4 – Συνδεσμολογία Arduino Nano με DHT22

Στην Εικόνα 7.5 εμφανίζεται η συνδεσμολογία της πλατφόρμας Arduino nano με το ρολόι πραγματικού χρόνου RTC3231.



Εικόνα 7.5 – Συνδεσμολογία Arduino Nano με RTC3231

Στην Εικόνα 7.6 εμφανίζεται η συνδεσμολογία της πλατφόρμας Arduino nano με τον ραδιοφωνικό πομποδέκτη nRF24L01.



Εικόνα 7.6 – Συνδεσμολογία Arduino Nano με nRF24L01

ΚΕΦΑΛΑΙΟ 8

ΗΛΕΚΤΡΟΝΙΚΗ ΠΛΑΤΦΟΡΜΑ THINGSPEAK

8.1 Περιγραφή του THINGSPEAK

Το ThingSpeak ™ είναι μια πλατφόρμα υπηρεσίας Analytics IoT (Internet of Things) που επιτρέπει να συγκεντρώνουμε, να απεικονίζουμε και να αναλύουμε ζωντανές ροές δεδομένων στο cloud. Το ThingSpeak παρέχει άμεσες απεικονίσεις δεδομένων που έχουν αναρτηθεί από τις συσκευές που είναι συνδεδεμένες σε αυτό. Με τη δυνατότητα εκτέλεσης κώδικα MATLAB® στο ThingSpeak μπορεί να εκτελεστεί ηλεκτρονική ανάλυση και επεξεργασία των δεδομένων, καθώς αυτά έρχονται.

Ορισμένες από τις βασικές δυνατότητες του ThingSpeak είναι οι παρακάτω:

• Εύκολη διαμόρφωση συσκευών για την αποστολή δεδομένων στο ThingSpeak χρησιμοποιώντας δημοφιλή πρωτόκολλα IoT.

• Προβολή των δεδομένων αισθητήρων σε πραγματικό χρόνο.

• Εμφάνιση συγκεντρωτικών στοιχείων κατ 'απαίτηση από τρίτες πηγές.

• Χρήση του MATLAB για την καλύτερη κατανόηση των δεδομένων του IoT.

• Αυτόματη εκτέλεση αναλυτικών στοιχείων του IoT με βάση χρονοδιαγράμματα ή συμβάντα.

• Δημιουργία συστημάτων IoT χωρίς εγκατάσταση διακομιστών ή ανάπτυξη λογισμικού ιστού.

• Αυτόματη επεξεργασία των δεδομένων και επικοινωνία με υπηρεσίες τρίτων όπως το Twitter®.

8.2 Συλλογή Δεδομένων

Αισθητήρες υπάρχουν παντού - στα σπίτια μας, τα έξυπνα τηλέφωνα, τα αυτοκίνητα, την υποδομή των πόλεων και τον βιομηχανικό εξοπλισμό. Οι αισθητήρες αυτοί ανιχνεύουν και μετρούν πληροφορίες σχετικά με όλα τα είδη θεμάτων όπως η θερμοκρασία, η υγρασία και η πίεση και επικοινωνούν αυτά τα δεδομένα με κάποια μορφή, όπως μια αριθμητική τιμή ή ένα ηλεκτρικό σήμα.

Οι αισθητήρες, αισθάνονται τα δεδομένα και συνήθως δρουν τοπικά. Το ThingSpeak επιτρέπει στους αισθητήρες, τα όργανα και τους ιστότοπους να στέλνουν δεδομένα στο cloud και να αποθηκεύονται είτε σε κάποιο ιδιωτικό είτε σε κάποιο δημόσιο κανάλι. Το ThingSpeak αποθηκεύει τα δεδομένα σε ιδιωτικά κανάλια από προεπιλογή, αλλά τα δημόσια κανάλια μπορούν να χρησιμοποιηθούν για την κοινή χρήση δεδομένων με άλλους. Μόλις τα δεδομένα βρίσκονται σε ένα κανάλι ThingSpeak, μπορούν να υποστούν διαφόρων ειδών επεξεργασία.

8.3 Ανάλυση Δεδομένων

Η αποθήκευση δεδομένων στο σύννεφο παρέχει εύκολη πρόσβαση στα δεδομένα. Χρησιμοποιώντας ηλεκτρονικά εργαλεία ανάλυσης, τα δεδομένα αυτά μπορούν να εξερευνηθούν και να απεικονιστούν. Μπορούν να ανακαλυφθούν σχέσεις μεταξύ των δεδομένων, πρότυπα, μοτίβα να υπολογιστούν νέα δεδομένα και να απεικονιστούν αντίστοιχα σε διάφορες μορφές γραφημάτων.

Το ThingSpeak παρέχει πρόσβαση στο MATLAB και βοηθάει στο να αποκτήσουν νόημα τα δεδομένα.

- Μετατροπή, συνδυασμός και υπολογισμός νέων δεδομένων
- Προγραμματισμός υπολογισμών για εκτέλεση σε συγκεκριμένες ώρες
- Οπτική κατανόηση των σχέσεων στα δεδομένα χρησιμοποιώντας ενσωματωμένες λειτουργίες σχεδίασης
- Συνδυασμός δεδομένων από πολλαπλά κανάλια για την δημιουργία μιας πιο εξελιγμένης ανάλυσης

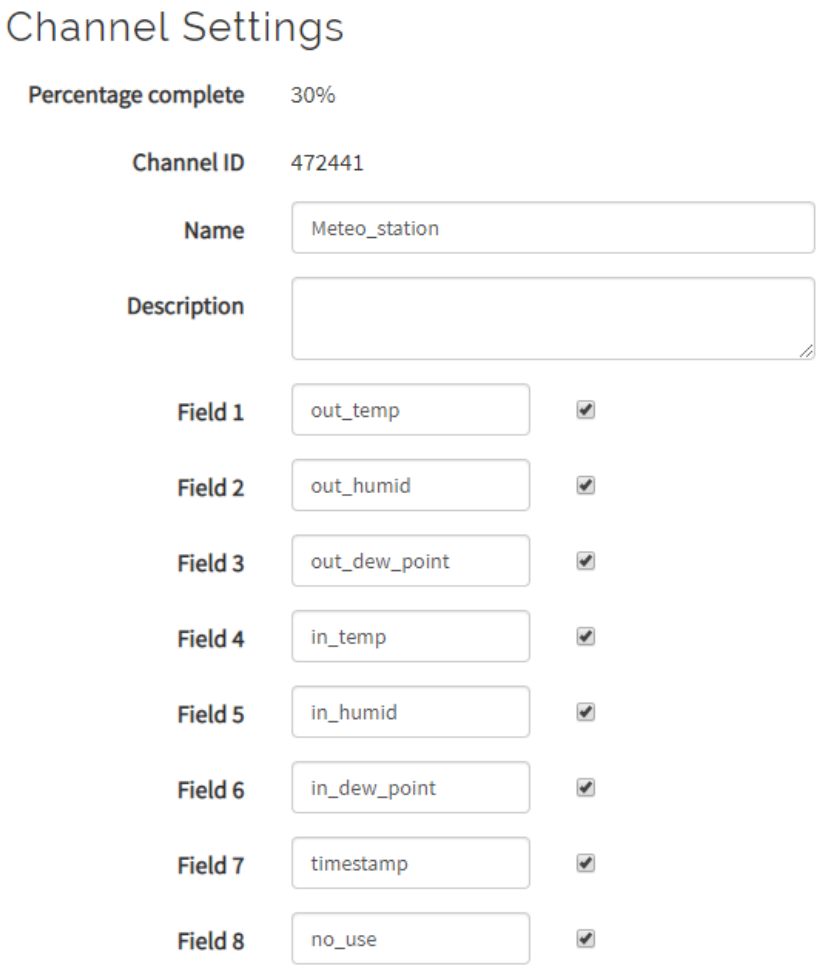
8.4 Χρήση Δεδομένων

Το ThingSpeak παρέχει εργαλεία που επιτρέπουν την επικοινωνία συσκευών για διάφορες ενέργειες. Για παράδειγμα υπάρχει η δυνατότητα να ενεργοποιηθεί μια απομακρυσμένη συσκευή όταν τα εισερχόμενα δεδομένα ξεπεράσουν μια συγκεκριμένη τιμή. Η λειτουργία που έχει χρησιμοποιηθεί στα πλαίσια της παρούσας εργασίας, είναι η προώθηση των δεδομένων από διάφορους αισθητήρες σε ένα πρόγραμμα συλλογής δεδομένων, που είναι εγκατεστημένο σε μια συσκευή Android.

8.5 Αξιοποίηση του ThingSpeak στην εργασία

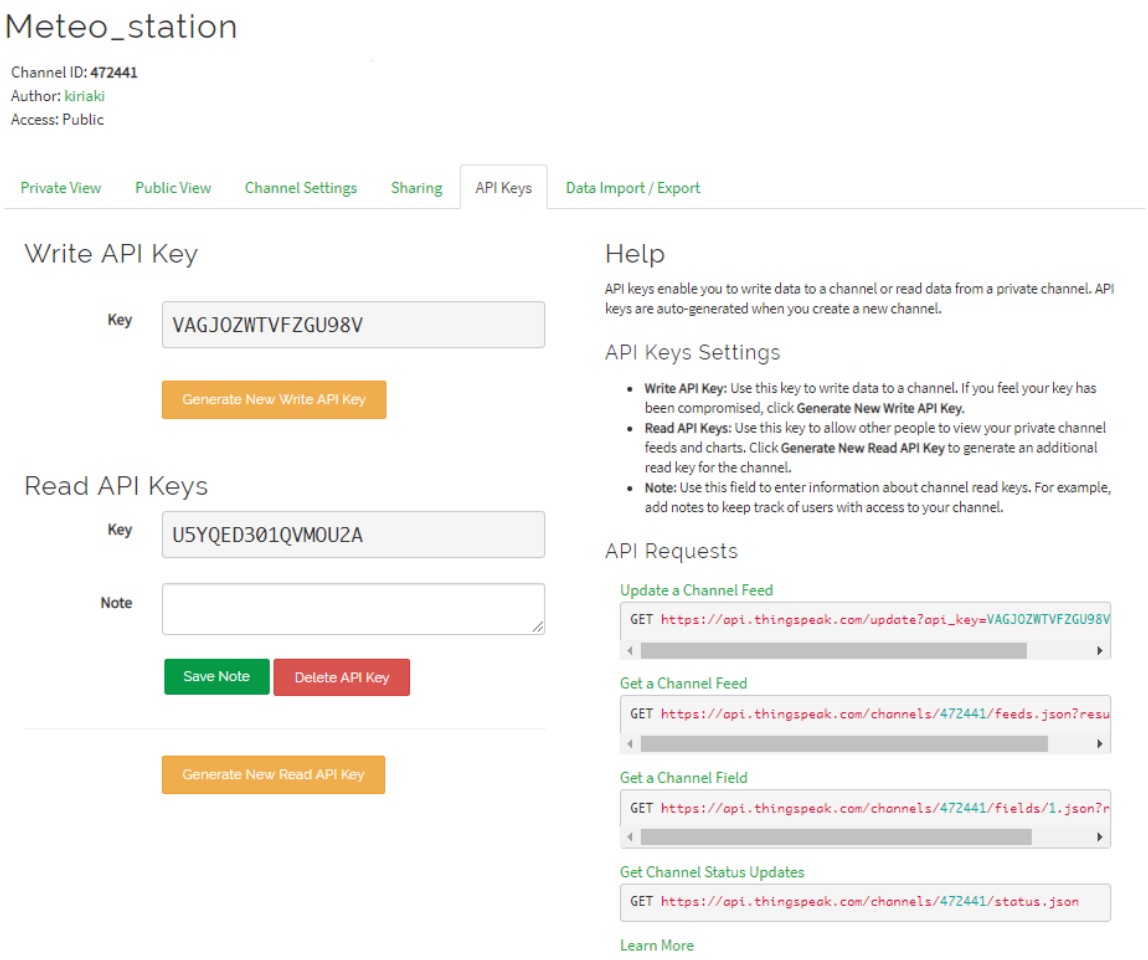
Στην παρούσα εργασία το ThingSpeak χρησιμοποιήθηκε για την δημιουργία ενός online καναλιού, το οποίο ενημερώνεται ανά τακτά χρονικά διαστήματα από δύο σετ αισθητήρων που έχουν τοποθετηθεί σε δύο διαφορετικά σημεία στον χώρο και καταγράφουν την θερμοκρασία και την υγρασία του περιβάλλοντος. Το κανάλι που δημιουργήθηκε ονομάστηκε Meteo_Station και του δόθηκε ένας μοναδικός αναγνωριστικός αριθμός ID. Το ThingSpeak δίνει την δυνατότητα αποθήκευσης 8 διαφορετικών πεδίων μέσα στο κανάλι. Στο συγκεκριμένο κανάλι χρησιμοποιήθηκαν τα 7 από τα 8 πεδία για αποθήκευση πληροφοριών. Συγκεκριμένα τα πεδία τα οποία θα καταγράφονται στο κανάλι είναι :

- Εξωτερική Θερμοκρασία
- Εξωτερική Υγρασία
- Εξωτερικό Σημείο Δρόσου
- Ημερομηνία Λήψης μέτρησης
- Εσωτερική Θερμοκρασία
- Εσωτερική Υγρασία
- Εσωτερικό Σημείο Δρόσου



Εικόνα 8.1 – Χαρακτηριστικά καναλιού ThingSpeak

Ένα από τα μεγαλύτερα πλεονεκτήματα του ThingSpeak είναι η παροχή 2 API (Application Programming Interface) κλειδιών, με τα οποία μπορεί κάποιος να ενημερώσει το κανάλι του ή να χρησιμοποιήσει τα δεδομένα του καναλιού του σε κάποια άλλη εφαρμογή. Επίσης το ThingSpeak προσφέρει κάποια βασικά παραδείγματα API requests για την ενημέρωση πληροφοριών ή την ανάκτηση πληροφοριών από το κανάλι.



Εικόνα 8.2 – API κλειδιά καναλιού ThingSpeak

Τέλος, το ThingSpeak έχει μια ακόμα δυνατότητα που αξιοποιήθηκε στην παρούσα εργασία. Την οπτική απεικόνιση των μετρήσεων που αποθηκεύονται στο κανάλι. Προσφέρει την δυνατότητα προσθήκης διαγραμμάτων, που απεικονίζουν τις μετρήσεις κάθε πεδίου ξεχωριστά, σε συνάρτηση με τον χρόνο που πάρθηκε η μέτρηση. Αυτό βοηθάει στην πιο εύκολη παρακολούθηση των μετρήσεων και της εξέλιξής τους. Στην Εικόνα 8.3 εμφανίζονται τα διαγράμματα που έχουν δημιουργηθεί από ένα χρονικό διάστημα μετρήσεων.

Εικόνα 8.3 – Διαγράμματα κάθε πεδίου στο ThingSpeak

ΚΕΦΑΛΑΙΟ 9

ANDROID ΚΑΙ ANDROID STUDIO

9.1 Περιγραφή των Android και Android Studio

Το Android είναι λειτουργικό σύστημα για συσκευές κινητής τηλεφωνίας το οποίο τρέχει τον πυρήνα του λειτουργικού Linux. Επιτρέπει στους κατασκευαστές λογισμικού να συνθέτουν κώδικα με την χρήση της γλώσσας προγραμματισμού Java, ελέγχοντας την συσκευή μέσω βιβλιοθηκών λογισμικού ανεπτυγμένων από την Google.

Το Android είναι κατά κύριο λόγο σχεδιασμένο για συσκευές με οθόνη αφής, όπως τα έξυπνα τηλέφωνα και τα τάμπλετ, με διαφορετικό περιβάλλον χρήσης για τηλεοράσεις (Android TV), αυτοκίνητα (Android Auto) και ρολόγια χειρός (Android Wear). Παρόλο που έχει αναπτυχθεί για συσκευές με οθόνη αφής, έχει χρησιμοποιηθεί σε κονσόλες παιχνιδιών, ψηφιακές φωτογραφικές μηχανές, συνηθισμένους Η/Υ και σε άλλες ηλεκτρονικές συσκευές.

Το Android Studio είναι ένα ολοκληρωμένο προγραμματιστικό περιβάλλον (IDE) για ανάπτυξη εφαρμογών στην πλατφόρμα Android.

9.2 Android και Οικιακός Μετεωρολογικός Σταθμός

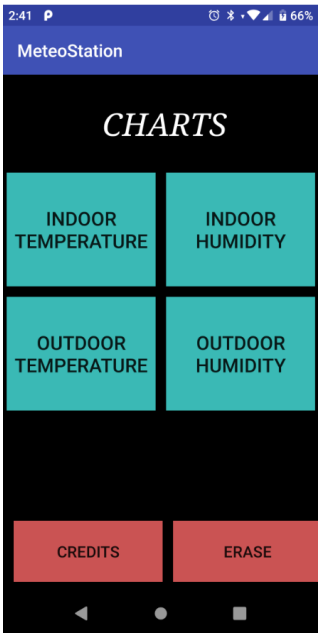
Στην παρούσα εργασία οι δυνατότητες του Android χρησιμοποιήθηκαν για την οπτική απεικόνιση των δεδομένων που συλλέχτηκαν από τον οικιακό μετεωρολογικό σταθμό. Για την ακρίβεια, δημιουργήθηκε μια εφαρμογή στην οποία μπορεί να έχει πρόσβαση οποιαδήποτε Android συσκευή, (τηλέφωνο, tablet, τηλεόραση) και η οποία απεικονίζει γραφήματα των μετρήσεων για την πιο εύκολη κατανόηση της συμπεριφοράς της θερμοκρασίας και της υγρασίας στους χώρους από τους οποίους έχουν παρθεί οι μετρήσεις.

Ξεκινώντας την εφαρμογή εμφανίζεται η αρχική οθόνη (Εικόνα 9.1), στην οποία έχει δημιουργηθεί ένα αναλογικό ρολόι για την απεικόνιση της ώρας και ακριβώς από κάτω η τρέχουσα ημερομηνία. Επίσης εμφανίζονται οι πιο πρόσφατες τιμές της θερμοκρασίας και της υγρασίας από τον αισθητήρα που έχει τοποθετηθεί μέσα στο σπίτι. Τέλος εμφανίζεται ένα κουμπί “MENU” που μεταφέρει τον χρήστη στην κεντρική οθόνη της εφαρμογής.

Εικόνα 9.1 – Αρχική Οθόνη Εφαρμογής

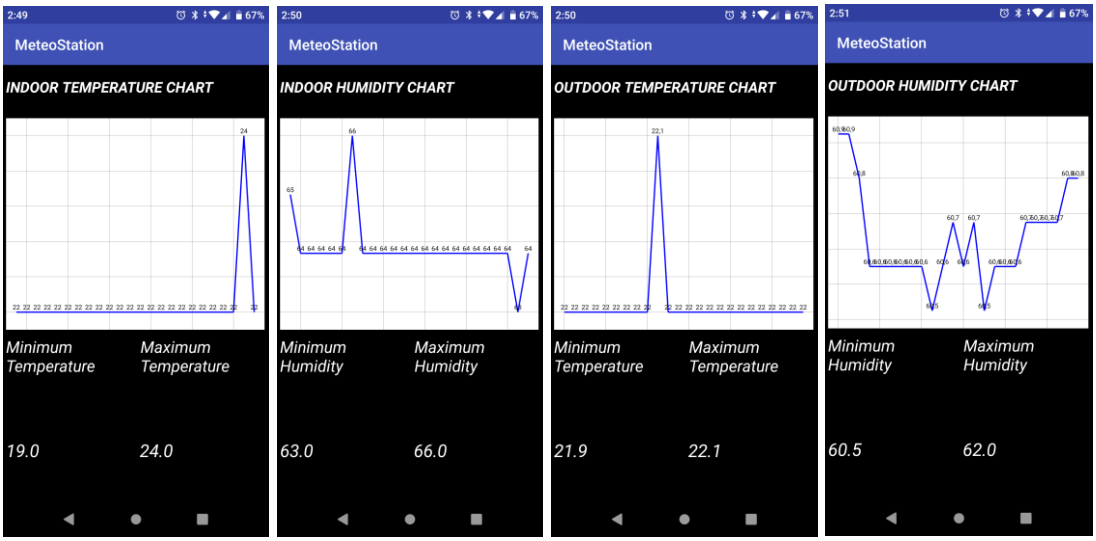
Στην κεντρική οθόνη της εφαρμογής (Εικόνα 9.2), ο χρήστης μπορεί να επιλέξει να δει ένα από τα τέσσερα διαθέσιμα γραφήματα που αφορούν:

- Την εσωτερική θερμοκρασία (Indoor Temperature)
- Την εσωτερική υγρασία (Indoor Humidity)
- Την εξωτερική θερμοκρασία (Outdoor Temperature)
- Την εξωτερική υγρασία (Outdoor Humidity)



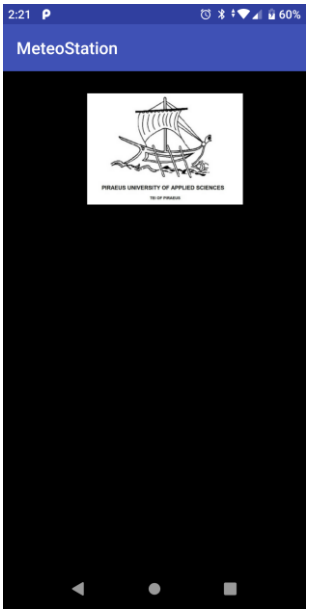
Εικόνα 9.2 – Κεντρική Οθόνη Εφαρμογής

Κάθε επιλογή αντιστοιχεί σε μια καινούργια οθόνη της εφαρμογής, η οποία περιέχει το κατάλληλο γράφημα και πληροφορίες για την μέγιστη και ελάχιστη τιμή του πεδίου μέτρησης που έχει επιλεγεί. Όλα τα δεδομένα συλλέγονται από το ηλεκτρονικό κανάλι ThingSpeak στο οποίο έχουν αποθηκευτεί πρωτύτερα και με την κατάλληλη επεξεργασία, εμφανίζονται τα επιθυμητά αποτελέσματα. Η Εικόνα 9.3 δείχνει μια τυχαία απεικόνιση της κάθε επιλογής.



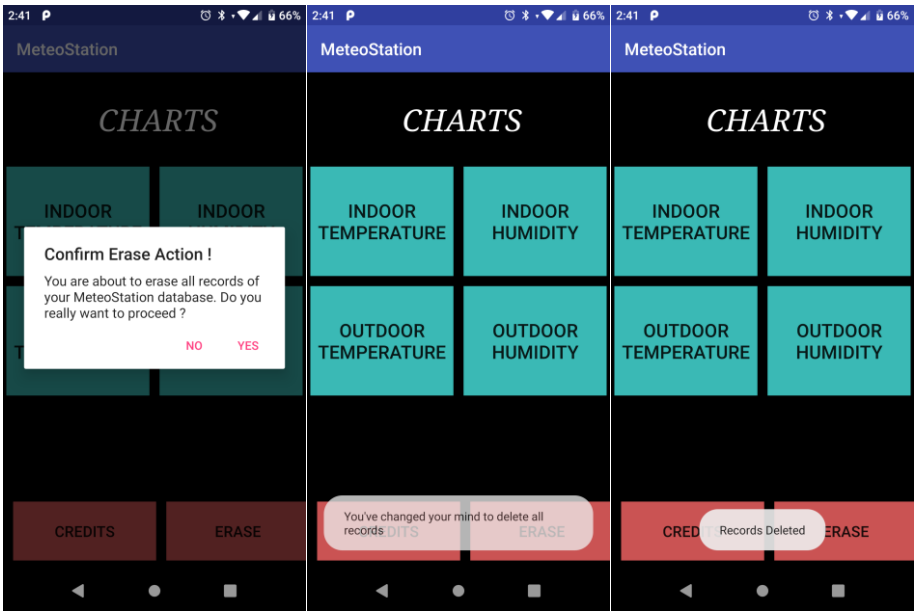
Εικόνα 9.3 – Οθόνες Γραφημάτων

Στην κεντρική οθόνη της εφαρμογής υπάρχουν ακόμα δύο επιλογές. Η μία επιλογή είναι η “CREDITS” (Εικόνα 9.4) στην οποία υπάρχουν οι προσωπικές πληροφορίες του προγραμματιστή.



Εικόνα 9.4 – Οθόνη CREDITS

Η άλλη επιλογή είναι η “ERASE” (Εικόνα 9.5) η οποία δίνει την δυνατότητα στον χρήστη να διαγράψει όλα τα δεδομένα που έχουν αποθηκευτεί στο κανάλι και να ξεκινήσουν οι μετρήσεις από την αρχή. Η επιλογή αυτή δόθηκε στον χρήστη για την περίπτωση που θέλει να αλλάξει τοποθεσία στον μετεωρολογικό σταθμό και θα ήθελε τα δεδομένα του να είναι καθαρά από μία τοποθεσία. Να σημειωθεί πως η επιλογή “ERASE” δεν αντιστοιχεί σε μια ακόμα οθόνη της εφαρμογής, αλλά σε μια εντολή η οποία θα φέρει το επιθυμητό αποτέλεσμα (Εικόνα 9.6 και Εικόνα 9.7).



Εικόνα 9.5 – Εικόνα 9.6 – Εικόνα 9.7 – Λειτουργία ERASE

Ο κώδικας της εφαρμογής αναπτύσσεται στο Παράρτημα Β’.

ΠΑΡΑΡΤΗΜΑ Α’

Στο παράρτημα αυτό παρατίθεται ο κώδικας ανάπτυξης του πομπού και του δέκτη του μετεωρολογικού σταθμού που δημιουργήθηκε στην πλατφόρμα Arduino IDE.

- ***Κώδικας ανάπτυξης πομπού (receiver)***

```
//-----------------------------------------------------------------------------------------------
//                                LIBRARIES
//-----------------------------------------------------------------------------------------------
#include <DHTesp.h> //Sensor library
#include <Wire.h> //I2C library
#include <RtcDS3231.h> //RTC library
#include "RF24.h" //nRF24L01 library
#include <ESP8266WiFi.h> //nodeMCU library
#include "ThingSpeak.h" //ThingSpeak library
#include "secrets.h" //Credential library

//-----------------------------------------------------------------------------------------------
//                                DEFINES
//-----------------------------------------------------------------------------------------------
int dhtPin = 02; //sensor pin

//-----------------------------------------------------------------------------------------------
//                                OBJECTS
//-----------------------------------------------------------------------------------------------
const char* server = "api.thingspeak.com"; //server name
WiFiClient client; //WiFiClient object
DHTesp DHT1; //sensor object
RtcDS3231<TwoWire> rtcObject(Wire); //real time clock object
RF24 myRadio (0, 15); //radio receiver object
byte addresses[][6] = {"0"};
String dateString;
String hours;
```

```
int minuteNow = 0;
int minutePrevious = 0;
struct package
{
  float temperature ;
  float humidity ;
};
typedef struct package Package;
Package data;
float previousIndoorHumidity = 0;
float previousIndoorTemperature = 0;
float previousRemoteHumidity = 0.1;
float previousRemoteTemperature = 0.1;
float indoorHumidity = 0;
float indoorTemperature = 0;
float remoteHumidity = 0.0;
float remoteTemperature = 0.0;
char datestring[20];

//------------------------------------------------------------------------------------------
//                                    SETUP
//------------------------------------------------------------------------------------------
void setup() {
  Serial.begin(9600);
  DHT1.setup(dhtPin, DHTesp::DHT11); //sets the sensor pin
  rtcObject.Begin();    //Starts I2C
  //setRTCTime(); //uncomment only the first time in order to set the clock

  //the below block connects to the wifi
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  Serial.print(".....");
  Serial.println();
```

```
  WiFi.begin(ssid, password);
  ThingSpeak.begin(client);  // Initialize ThingSpeak

  while (WiFi.status() != WL_CONNECTED) { //wait until the wifi is connected
    delay(500);
  }
  if (WiFi.status() != WL_CONNECTED) { //inform user if wifi is connected
    Serial.println("WiFi not connected");
    Serial.println();
  }else {
    Serial.println("WiFi connected");
    Serial.println();
  }
  delay(2000);
  startWirelessCommunication();
}

void setRTCTime() //procedure to set the date and time in RTC
{
  RtcDateTime compiled = RtcDateTime(__DATE__, __TIME__);
  Serial.println("Date Time compiled");
  printDateTime(compiled);
  Serial.println();
  rtcObject.SetDateTime(compiled); //configure the RTC with object
  if (!rtcObject.GetIsRunning())
  {
    Serial.println("RTC was not actively running, starting now");
    rtcObject.SetIsRunning(true);
  }
}

void loop() { //every one second the receiver checks for data from transmitter
  checkForWirelessData();
  delay(1000);
```

```
}

void startWirelessCommunication() //sets the radio of the receiver in the same range as the trasmitter
{
  myRadio.begin();
  myRadio.setChannel(115);
  myRadio.setPALevel(RF24_PA_MAX);
  myRadio.setDataRate( RF24_250KBPS ) ;
  myRadio.openReadingPipe(1, addresses[0]);
  myRadio.startListening();
  delay(100);
}

void checkForWirelessData()
{
  if ( myRadio.available())
  {
    while (myRadio.available()) //when the transmitter sends a package of data the old data are stored in variables
    {
      myRadio.read( &data, sizeof(data) );
      previousRemoteTemperature = remoteTemperature;
      previousRemoteHumidity = remoteHumidity;
      remoteTemperature = data.temperature;
      remoteHumidity = data.humidity;
    }
    getAndPrintTime(); //gets the timestamp of the measurement
    Serial.print("Outdoor Temperature: ");
    Serial.println(data.temperature);
    Serial.print("Outdoor Humidity: ");
    Serial.println(data.humidity);
    readSensor(); //reads the sensor
    sendToThingSpeak(); //updates channel
```

```
  }
}

void getAndPrintTime() //procedure to get the current timestamp
{
  delay(100);
  RtcDateTime now = rtcObject.GetDateTime();
  Serial.print("DateTime: ");
  printDateTime(now);
}

#define countof(a) (sizeof(a) / sizeof(a[0]))

void printDateTime(const RtcDateTime& dt) //procedure to print the timestamp in certain format
{
  snprintf_P(datestring,
             countof(datestring),
             PSTR("%02u/%02u/%04u %02u:%02u:%02u"),
             dt.Month(),
             dt.Day(),
             dt.Year(),
             dt.Hour(),
             dt.Minute(),
             dt.Second() );
  Serial.println(datestring);
}

void readSensor() { //procedure to read the sensor
  previousIndoorTemperature = indoorTemperature;
  previousIndoorHumidity = indoorHumidity;
  indoorHumidity = DHT1.getHumidity();
  indoorTemperature = DHT1.getTemperature();
```

```
  Serial.print("Indoor Temperature = ");
  Serial.println(DHT1.getTemperature());
  Serial.print("Indoor Humidity = ");
  Serial.println(DHT1.getHumidity());
  delay(1000);
}

void sendToThingSpeak() //procedure to update ThhingSpeak channel
{
  Serial.print("Start sending data to ThingSpeak");
  Serial.println();
  //calculate the outdoor and indoor dew point
  float in_gamma = log(indoorHumidity / 100) + ((17.62 * indoorTemperature) /
(243.5 + indoorTemperature));
  float in_dp = 243.5 * in_gamma / (17.62 - in_gamma);
  float out_gamma = log(remoteHumidity / 100) + ((17.62 * remoteTemperature) /
(243.5 + remoteTemperature));
  float out_dp = 243.5 * out_gamma / (17.62 - out_gamma);

  // set the fields with the corresponding values
  ThingSpeak.setField(1, remoteTemperature);
  ThingSpeak.setField(2, remoteHumidity);
  ThingSpeak.setField(3, out_dp);
  ThingSpeak.setField(4, indoorTemperature);
  ThingSpeak.setField(5, indoorHumidity);
  ThingSpeak.setField(6, in_dp);
  ThingSpeak.setField(7, datestring);

  // write to the ThingSpeak channel
  int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
  if(x == 200){
    Serial.println("Channel update successful.");
  }
  else{
```

```
    Serial.println("Problem updating channel. HTTP error code " + String(x));
  }
  // thingspeak needs minimum 15 sec delay between updates. Our delay is 20 sec
  delay(20000);
}
```

- ***Κώδικας ανάπτυξης δέκτη (transmitter)***

```
//-----------------------------------------------------------------------------------------------
//                              LIBRARIES
//-----------------------------------------------------------------------------------------------
#include <avr/sleep.h>//this AVR library contains the methods that controls the sleep modes
#include "DHT.h" //sensor library
#include <SPI.h> //serial port communication library
#include "RF24.h" //radio transmitter library
#include <DS3232RTC.h>  //RTC Library

//-----------------------------------------------------------------------------------------------
//                              DEFINES
//-----------------------------------------------------------------------------------------------
#define DHTPIN 4 //pin that is connected to the sensor
#define DHTTYPE DHT22 //sensor model
#define interruptPin 2 //Pin that is used to wake up the Arduino

//-----------------------------------------------------------------------------------------------
//                              OBJECTS
//-----------------------------------------------------------------------------------------------
RF24 myRadio (7, 8);
byte addresses[][6] = {"0"};
const int led_pin = 5; //led pin for usage monitoring
unsigned long seconds = 1000L;
unsigned long minutes = seconds * 5;
const int time_interval = 30; // Sets the wakeup intervall in minutes

struct package
{
  float temperature ;
  float humidity ;
};
typedef struct package Package;
Package data;
DHT dht(DHTPIN, DHTTYPE);
```

```
void setup()
{
  Serial.begin(9600);
  pinMode(led_pin, OUTPUT); //set led pin as an output
  pinMode(interruptPin, INPUT_PULLUP); //Set pin d2 to input using the buildin pullup resistor
  digitalWrite(led_pin, HIGH); // Flash a light to show transmitting

 // initialize the alarms to known values, clear the alarm flags, clear the alarm interrupt flags
  RTC.setAlarm(ALM1_MATCH_DATE, 0, 0, 0, 1);
  RTC.setAlarm(ALM2_MATCH_DATE, 0, 0, 0, 1);
  RTC.alarm(ALARM_1);
  RTC.alarm(ALARM_2);
  RTC.alarmInterrupt(ALARM_1, false);
  RTC.alarmInterrupt(ALARM_2, false);
  RTC.squareWave(SQWAVE_NONE);
  //
  /*
     The below block is used once in order to set the time on the RTC.
  */
  /*Begin block
  tmElements_t tm;
  tm.Hour = 22;               // set the RTC to an arbitrary time
  tm.Minute = 00;
  tm.Second = 00;
  tm.Day = 12;
  tm.Month = 5;
  tm.Year = 2019 - 1970;      // tmElements_t.Year is the offset from 1970
  RTC.write(tm);              // set the RTC from the tm structure
  //Block end * */
  time_t t; //create a temporary time variable to set and read the time from the RTC
  t = RTC.get(); //Get the current time of the RTC
  RTC.setAlarm(ALM1_MATCH_SECONDS , 0, second(t) + time_interval, 0, 0); // Setting alarm 30 SECS to go off 1 minute from now
  RTC.alarm(ALARM_1); // clear the alarm flag
```

```
  RTC.squareWave(SQWAVE_NONE); // configure the INT/SQW pin for "interrupt" operation
  RTC.alarmInterrupt(ALARM_1, true);  // enable interrupt output for Alarm 1
  dht.begin(); //set sensor up
  //set channel for communication
  myRadio.begin();
  myRadio.setChannel(115);
  myRadio.setPALevel(RF24_PA_MAX);
  myRadio.setDataRate( RF24_250KBPS ) ;
  myRadio.openWritingPipe( addresses[0]);
  delay(1000);
  //
}

void loop() {
  delay(5000); //wait 5 seconds before going to sleep
  Going_To_Sleep();
}

void Going_To_Sleep() {
  sleep_enable(); //Enabling sleep mode
  attachInterrupt(0, wakeUp, LOW); //attaching a interrupt to pin d2
  set_sleep_mode(SLEEP_MODE_PWR_DOWN);//Setting the sleep mode, in our case full sleep
  digitalWrite(led_pin, LOW); //turning LED off
  time_t t;// create temp time variable
  t = RTC.get(); //get current time from rtc
  Serial.println("Sleep    Time: " + String(hour(t)) + ":" + String(minute(t)) + ":" + String(second(t))); //printstime stamp on serial monitor
  delay(1000); //wait a second to allow the led to be turned off before going to sleep
  sleep_cpu(); //activating sleep mode
  //next line of code executed after the interrupt
  Serial.println("just woke up!");
  digitalWrite(led_pin, HIGH); //turning LED on
  temp_Humi();//function that reads the temp and the humidity
  t = RTC.get(); //get the current timestamp from RTC
```

```
  Serial.println("WakeUp Time: " + String(hour(t)) + ":" + String(minute(t)) + ":" +
String(second(t))); //Print time stamp
  //Set New Alarm
  RTC.setAlarm(ALM1_MATCH_SECONDS , 0, second(t) + time_interval, 0, 0);
  // clear the alarm flag
  RTC.alarm(ALARM_1);
}

void wakeUp() {
  Serial.println("Interrrupt Fired"); //Print message to serial monitor
  sleep_disable(); //Disable sleep mode
  detachInterrupt(0); //Remove the interrupt from pin 2;
}

void temp_Humi()
{
  readSensor();
  Serial.println(data.humidity); //print humidity to serial monitor
  Serial.println(data.temperature); //print temperature to serial monitor
  myRadio.write(&data, sizeof(data)); //pass data to channel for transmission
}

void readSensor()
{
  data.humidity = dht.readHumidity();
  data.temperature = dht.readTemperature();
}
```

ΠΑΡΑΡΤΗΜΑ Β’

Στο παράρτημα αυτό παρατίθεται ο βασικός κώδικας ανάπτυξης της εφαρμογής του μετεωρολογικού σταθμού που δημιουργήθηκε στην πλατφόρμα Android Studio.

- MainMenu.java

```
package com.example.meteostation;

import android.content.DialogInterface;
import android.content.Intent;
import android.os.AsyncTask;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.Gravity;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

public class MainMenu extends AppCompatActivity {

    static final String USER_API_KEY = "VMB39JSXN8RY4T04";
    static final String API_URL =
"https://api.thingspeak.com/channels/472441/feeds.json?api_key=";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main_menu);

        Button inTempButton = findViewById(R.id.inTempChart);
        inTempButton.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                toInTemp();
            }

        });

        Button outTempButton = findViewById(R.id.outTempChart);
        outTempButton.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                toOutTemp();
```

```
            }

        });

        Button inHumidButton = findViewById(R.id.inHumidChart);
        inHumidButton.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                toInHumid();
            }

        });

        Button outHumidButton = findViewById(R.id.outHumidChart);
        outHumidButton.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                toOutHumid();
            }

        });

        Button creditsButton = findViewById(R.id.credits);
        creditsButton.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                toCredits();
            }

        });

        Button eraseAllButton = findViewById(R.id.eraseAll);
        eraseAllButton.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                toErase();
            }

        });
    }
    public void toInTemp(){
        Intent intent = new Intent(this, TempChart.class);
        intent.putExtra("place","in");
        startActivity(intent);

    }
    public void toOutTemp(){
        Intent intent = new Intent(this, TempChart.class);
        intent.putExtra("place","out");
        startActivity(intent);
    }
    public void toInHumid(){
        Intent intent = new Intent(this, HumidChart.class);
        intent.putExtra("place","in");
        startActivity(intent);
    }
    public void toOutHumid(){
```

```
        Intent intent = new Intent(this, HumidChart.class);
        intent.putExtra("place","out");
        startActivity(intent);
    }
    public void toCredits(){
        Intent intent = new Intent(this, Credits.class);
        startActivity(intent);
    }
    public void toErase(){
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Confirm Erase Action !");
        builder.setMessage("You are about to erase all records of your
MeteoStation database. Do you really want to proceed ?");
        builder.setCancelable(false);
        builder.setPositiveButton("Yes", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(getApplicationContext(), "You've
choosen to delete all records", Toast.LENGTH_SHORT).show();
                new RetrieveFeedTask().execute();
            }
        });

        builder.setNegativeButton("No", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(getApplicationContext(), "You've
changed your mind to delete all records", Toast.LENGTH_SHORT).show();
            }
        });

        builder.show();
    }
    class RetrieveFeedTask extends AsyncTask<Void, Void, String> {


        protected String doInBackground(Void... urls) {
            try {
                URL url = new URL(API_URL + USER_API_KEY);
                HttpURLConnection urlConnection = (HttpURLConnection)
url.openConnection();
                urlConnection.setDoOutput(true);
                urlConnection.setRequestProperty("Content-Type",
                        "application/x-www-form-urlencoded");
                urlConnection.setRequestMethod("DELETE");
                urlConnection.connect();
                urlConnection.disconnect();
                return "Records Deleted";
            } catch (Exception e) {
                Log.e("ERROR", e.getMessage(), e);
                return null;
            }
        }

        protected void onPostExecute(String response) {
            if (response == null) {
                Toast.makeText(getApplicationContext(), "Something
went wrong", Toast.LENGTH_SHORT).show();
            }else{
```

```
                    Toast.makeText(getApplicationContext(), "Records
Deleted", Toast.LENGTH_SHORT).show();
                }
            }
        }

}
```

- ClockView.java

```
package com.example.meteostation;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Rect;
import android.util.AttributeSet;
import android.util.TypedValue;
import java.util.Calendar;
import android.view.View;

public class ClockView extends View {
    private int height, width = 0;
    private int padding = 0;
    private int fontSize = 0;
    private int numeralSpacing = 0;
    private int handTruncation, hourHandTruncation = 0;
    private int radius = 0;
    private Paint paint;
    private boolean isInit;
    private int[] numbers = {1,2,3,4,5,6,7,8,9,10,11,12};
    private Rect rect = new Rect();

    public ClockView(Context context) {
        super(context);
    }

    public ClockView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    public ClockView(Context context, AttributeSet attrs, int
defStyleAttr) {
        super(context, attrs, defStyleAttr);
    }

    private void initClock() {
        height = getHeight();
        width = getWidth();
        padding = numeralSpacing + 50;
        fontSize = (int)
TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_SP, 13,
                getResources().getDisplayMetrics());
        int min = Math.min(height, width);
        radius = min / 2 - padding;
        handTruncation = min / 20;
        hourHandTruncation = min / 7;
        paint = new Paint();
        isInit = true;
    }

    @Override
    protected void onDraw(Canvas canvas) {
        if (!isInit) {
            initClock();
        }

        canvas.drawColor(Color.BLACK);
```

```
            drawCircle(canvas);
            drawCenter(canvas);
            drawNumeral(canvas);
            drawHands(canvas);

            postInvalidateDelayed(500);
            invalidate();
        }

        private void drawHand(Canvas canvas, double loc, boolean isHour)
    {
            double angle = Math.PI * loc / 30 - Math.PI / 2;
            int handRadius = isHour ? radius - handTruncation -
    hourHandTruncation : radius - handTruncation;
            canvas.drawLine(width / 2, height / 2,
                    (float) (width / 2 + Math.cos(angle) * handRadius),
                    (float) (height / 2 + Math.sin(angle) * handRadius),
                    paint);
        }

        private void drawHands(Canvas canvas) {
            Calendar c = Calendar.getInstance();
            float hour = c.get(Calendar.HOUR_OF_DAY);
            hour = hour > 12 ? hour - 12 : hour;
            drawHand(canvas, (hour + c.get(Calendar.MINUTE) / 60) * 5f,
    true);
            drawHand(canvas, c.get(Calendar.MINUTE), false);
            drawHand(canvas, c.get(Calendar.SECOND), false);
        }

        private void drawNumeral(Canvas canvas) {
            paint.setTextSize(fontSize);

            for (int number : numbers) {
                String tmp = String.valueOf(number);
                paint.getTextBounds(tmp, 0, tmp.length(), rect);
                double angle = Math.PI / 6 * (number - 3);
                int x = (int) (width / 2 + Math.cos(angle) * radius -
    rect.width() / 2);
                int y = (int) (height / 2 + Math.sin(angle) * radius +
    rect.height() / 2);
                canvas.drawText(tmp, x, y, paint);
            }
        }

        private void drawCenter(Canvas canvas) {
            paint.setStyle(Paint.Style.FILL);
            canvas.drawCircle(width / 2, height / 2, 12, paint);
        }

        private void drawCircle(Canvas canvas) {
            paint.reset();

    paint.setColor(getResources().getColor(android.R.color.white));
            paint.setStrokeWidth(5);
            paint.setStyle(Paint.Style.STROKE);
            paint.setAntiAlias(true);
            canvas.drawCircle(width / 2, height / 2, radius + padding -
    10, paint);
        }
    }
```

- MainActivity.java

```
package com.example.meteostation;

import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.os.AsyncTask;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.Html;
import android.util.Log;
import android.view.Gravity;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.github.mikephil.charting.charts.LineChart;
import com.github.mikephil.charting.data.Entry;
import com.github.mikephil.charting.data.LineData;
import com.github.mikephil.charting.data.LineDataSet;
import com.github.mikephil.charting.interfaces.datasets.ILineDataSet;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import org.w3c.dom.Text;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Locale;

public class MainActivity extends AppCompatActivity {

    TextView nowDate;
    TextView nowTemp;
    TextView nowHumid;
    int last_entry_id, feeds_length;
    double in_temp, in_humid;

    static final String API_KEY = "U5YQED301QVMOU2A";
    static final String API_URL =
"https://api.thingspeak.com/channels/472441/feeds.json?api_key=";
    public String jsonResponse;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        nowTemp = findViewById(R.id.nowTemp);
        nowHumid = findViewById(R.id.nowHumid);
```

```
        Button mainMenuButton = findViewById(R.id.button);

        mainMenuButton.setOnClickListener(new View.OnClickListener()
{

            @Override
            public void onClick(View v) {
                toMainMenu();
            }

        });
        TextView nowDate = findViewById(R.id.nowDate);

        SimpleDateFormat dateF = new SimpleDateFormat("EEEE, d MMMM
yyyy", Locale.getDefault());
        String date = dateF.format(Calendar.getInstance().getTime());

        nowDate.setText(date);
        nowDate.setGravity(Gravity.CENTER);
        new RetrieveFeedTask().execute();


    }
    class RetrieveFeedTask extends AsyncTask<Void, Void, String> {


        protected String doInBackground(Void... urls) {

            try {
                URL url = new URL(API_URL + API_KEY);
                HttpURLConnection urlConnection = (HttpURLConnection)
url.openConnection();
                try {
                    BufferedReader bufferedReader = new
BufferedReader(new
InputStreamReader(urlConnection.getInputStream()));
                    StringBuilder stringBuilder = new
StringBuilder();
                    String line;
                    while ((line = bufferedReader.readLine()) !=
null) {
                        stringBuilder.append(line).append("\n");
                    }
                    jsonResponse = stringBuilder.toString();
                    bufferedReader.close();
                    return stringBuilder.toString();
                }
                finally{
                    urlConnection.disconnect();
                }
            }
            catch(Exception e) {
                Log.e("ERROR", e.getMessage(), e);
                return null;
            }
        }

        protected void onPostExecute(String response) {
            if(response == null) {
                response = "THERE WAS AN ERROR";
            }else {
```

```
                try {
                    JSONObject reader = new JSONObject(jsonResponse);

                    JSONObject channel =
reader.getJSONObject("channel");
                    last_entry_id = channel.getInt("last_entry_id");
                    // Getting JSON Array node
                    JSONArray feeds = reader.getJSONArray("feeds");
                    feeds_length = feeds.length();
                    // looping through All Feeds
                    for (int i = 0; i < feeds.length(); i++) {
                        JSONObject feed = feeds.getJSONObject(i);
                        int id = feed.getInt("entry_id");
                        if (id == last_entry_id){
                            in_temp = feed.getDouble("field4");
                            in_humid = feed.getDouble("field5");
                        }else{
                            in_temp = 0;
                            in_humid = 0;
                        }
                    }
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }

            nowTemp.setText(String.valueOf(in_temp) + " " + "oC");
            nowHumid.setText(String.valueOf(in_humid) + " " + "%");
            nowTemp.setGravity(Gravity.CENTER);
            nowHumid.setGravity(Gravity.CENTER);
        }
    }

    public void toMainMenu(){
        Intent intent = new Intent(this, MainMenu.class);
        startActivity(intent);
    }
}
```

- TempChart.java

```
package com.example.meteostation;

import android.graphics.Color;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

import com.github.mikephil.charting.charts.LineChart;
import com.github.mikephil.charting.data.Entry;
import com.github.mikephil.charting.data.LineData;
import com.github.mikephil.charting.data.LineDataSet;

import android.os.AsyncTask;
import android.util.Log;
import android.view.Gravity;
import android.widget.TextView;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

import java.text.DecimalFormat;
import java.util.ArrayList;

public class TempChart extends AppCompatActivity {

    LineChart TempChart;
    TextView minTemp;
    TextView maxTemp;
    TextView TempLabel;
    int last_entry_id;
    int feeds_length, feeds_start;
    String check_place;
    static final String API_KEY = "U5YQED301QVMOU2A";
    static final String API_URL =
"https://api.thingspeak.com/channels/472441/feeds.json?api_key=";
    ArrayList<MyFeeds> myFeeds = new ArrayList<MyFeeds>();
    public String jsonResponse;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_temp_chart);
        minTemp = findViewById(R.id.minTemp);
        maxTemp = findViewById(R.id.maxTemp);
        TempChart = findViewById(R.id.TempChart);
        TempLabel = findViewById(R.id.TempLabel );

        Bundle b = getIntent().getExtras();

        check_place = b.getString("place", "");

        Thread t = new Thread() {
            public void run() {
                for(;;)
```

```
                {
                    new RetrieveFeedTask().execute();
                    try {
                        Thread.sleep(3000);
                    } catch (Exception e) {
                        Log.e("Error: ", e.getMessage(), e);
                    }
                }
            }
        };
        t.start();

    }
    class RetrieveFeedTask extends AsyncTask<Void, Void, String> {


        protected String doInBackground(Void... urls) {

            try {
                URL url = new URL(API_URL + API_KEY);
                HttpURLConnection urlConnection = (HttpURLConnection)
url.openConnection();
                try {
                    BufferedReader bufferedReader = new
BufferedReader(new
InputStreamReader(urlConnection.getInputStream()));
                    StringBuilder stringBuilder = new
StringBuilder();
                    String line;
                    while ((line = bufferedReader.readLine()) !=
null) {
                        stringBuilder.append(line).append("\n");
                    }
                    jsonResponse = stringBuilder.toString();
                    bufferedReader.close();
                    return stringBuilder.toString();
                }
                finally{
                    urlConnection.disconnect();
                }
            }
            catch(Exception e) {
                Log.e("ERROR", e.getMessage(), e);
                return null;
            }
        }

        protected void onPostExecute(String response) {
            if(response == null) {
                response = "THERE WAS AN ERROR";
            }else {

                try {
                    JSONObject reader = new JSONObject(jsonResponse);

                    JSONObject channel =
reader.getJSONObject("channel");
                    last_entry_id = channel.getInt("last_entry_id");
                    // Getting JSON Array node
                    JSONArray feeds = reader.getJSONArray("feeds");
                    feeds_length = feeds.length();
```

```
                    feeds_start = feeds_length - 24;
                    if (feeds_start < 0) feeds_start = 0;
                    myFeeds.clear();
                    // looping through All Feeds
                    for (int i = 0; i < feeds_length; i++) {
                        JSONObject feed = feeds.getJSONObject(i);
                        int id = feed.getInt("entry_id");
                        double out_temp = feed.getDouble("field1");
                        double out_humid = feed.getDouble("field2");
                        double in_temp = feed.getDouble("field4");
                        double in_humid = feed.getDouble("field5");
                        String entry_time = feed.getString("field7");

                        myFeeds.add(new MyFeeds(id, out_temp,
out_humid, in_temp, in_humid, entry_time));

                    }
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }

            TempChart.setDrawGridBackground(false);
            // no description text
            TempChart.getDescription().setEnabled(false);
            // set an alternative background color
            TempChart.setBackgroundColor(Color.WHITE);
            TempChart.setViewPortOffsets(0f, 0f, 0f, 0f);
            TempChart.getXAxis().setSpaceMin(1);
            TempChart.getXAxis().setSpaceMax(1f);

            setData(feeds_length, check_place, TempChart);

            if (check_place.equals("in")){
                TempLabel.setText("INDOOR TEMPERATURE CHART");
            }else if (check_place.equals("out")){
                TempLabel.setText("OUTDOOR TEMPERATURE CHART");
            }
            minTemp.setText(getMinTemp(feeds_length, check_place));
            maxTemp.setText(getMaxTemp(feeds_length, check_place));
        }
    }

    private void setData(int count, String place, LineChart
tempChart) {

        ArrayList<Entry> yValues = new ArrayList<>();
        for (int i = feeds_start; i < count; i++) {
            MyFeeds addFeed = myFeeds.get(i);
            if (place.equals("in")) {
                yValues.add(new Entry(addFeed.getEntry_id(),
addFeed.getIn_temp()));
            }else if (place.equals("out")) {
                yValues.add(new Entry(addFeed.getEntry_id(),
addFeed.getOut_temp()));
            }
        }

        LineDataSet set1;

        // create a dataset and give it a type
```

```
            set1 = new LineDataSet(yValues, "Temperature");

            set1.setValueFormatter(new DecimalRemover(new
    DecimalFormat("###,###,###.##")));
            set1.setDrawCircles(false);
            set1.setLineWidth(1.8f);
            set1.setCircleRadius(4f);
            set1.setCircleColor(Color.BLUE);
            set1.setColor(Color.BLUE);
            set1.setFillColor(Color.BLUE);

            // create a data object with the datasets
            LineData data = new LineData();

    data.removeDataSet(data.getDataSetByLabel("Temperature",true));
            data.addDataSet(set1);
            data.setValueTextSize(9f);

            // set data
            tempChart.setData(data);
            tempChart.notifyDataSetChanged();
            tempChart.getLegend().setEnabled(false);
            tempChart.invalidate(); // refresh
        }
        private String getMaxTemp(int count, String place) {
            float maxTemp = (float) 0.0;
            for (int i = 0; i < count; i++) {
                MyFeeds addFeed = myFeeds.get(i);
                if (place.equals("in")) {
                    if (addFeed.getIn_temp() > maxTemp) {
                        maxTemp = addFeed.getIn_temp();
                    }
                }else if (place.equals("out")) {
                    if (addFeed.getOut_temp() > maxTemp) {
                        maxTemp = addFeed.getOut_temp();
                    }
                }
            }
            return Float.toString(maxTemp);
        }

        private String getMinTemp(int count, String place) {
            float minTemp = (float) 100.0;
            for (int i = 0; i < count; i++) {
                MyFeeds addFeed = myFeeds.get(i);
                if (place.equals("in")) {
                    if (addFeed.getIn_temp() < minTemp) {
                        minTemp = addFeed.getIn_temp();
                    }
                }else if (place.equals("out")) {
                    if (addFeed.getOut_temp() < minTemp) {
                        minTemp = addFeed.getOut_temp();
                    }
                }
            }
            return Float.toString(minTemp);
        }
    }
```

- HumidChart.java

```
package com.example.meteostation;

import android.graphics.Color;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

import com.github.mikephil.charting.charts.LineChart;
import com.github.mikephil.charting.data.Entry;
import com.github.mikephil.charting.data.LineData;
import com.github.mikephil.charting.data.LineDataSet;

import android.os.AsyncTask;
import android.util.Log;
import android.widget.TextView;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

import java.text.DecimalFormat;
import java.util.ArrayList;

public class HumidChart extends AppCompatActivity {

    LineChart HumidChart;
    TextView minHumid;
    TextView maxHumid;
    TextView HumidLabel;
    int last_entry_id;
    int feeds_length, feeds_start;
    String check_place;
    static final String API_KEY = "U5YQED301QVMOU2A";
    static final String API_URL =
"https://api.thingspeak.com/channels/472441/feeds.json?api_key=";
    ArrayList<MyFeeds> myFeeds = new ArrayList<MyFeeds>();
    public String jsonResponse;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_humid_chart);
        minHumid = findViewById(R.id.minHumid);
        maxHumid = findViewById(R.id.maxHumid);
        HumidChart = findViewById(R.id.HumidChart);
        HumidLabel = findViewById(R.id.HumidLabel );

        Bundle b = getIntent().getExtras();

        check_place = b.getString("place", "");

        Thread t = new Thread() {
            public void run() {
                for(;;)
                {
```

```
                    new RetrieveFeedTask().execute();
                    try {
                        Thread.sleep(3000);
                    } catch (Exception e) {
                        Log.e("Error: ", e.getMessage(), e);
                    }
                }
            }
        };
        t.start();

    }
    class RetrieveFeedTask extends AsyncTask<Void, Void, String> {


        protected String doInBackground(Void... urls) {

            try {
                URL url = new URL(API_URL + API_KEY);
                HttpURLConnection urlConnection = (HttpURLConnection)
url.openConnection();
                try {
                    BufferedReader bufferedReader = new
BufferedReader(new
InputStreamReader(urlConnection.getInputStream()));
                    StringBuilder stringBuilder = new
StringBuilder();
                    String line;
                    while ((line = bufferedReader.readLine()) !=
null) {
                        stringBuilder.append(line).append("\n");
                    }
                    jsonResponse = stringBuilder.toString();
                    bufferedReader.close();
                    return stringBuilder.toString();
                }
                finally{
                    urlConnection.disconnect();
                }
            }
            catch(Exception e) {
                Log.e("ERROR", e.getMessage(), e);
                return null;
            }
        }

        protected void onPostExecute(String response) {
            if(response == null) {
                response = "THERE WAS AN ERROR";
            }else{

                try {
                    JSONObject reader = new JSONObject(jsonResponse);

                    JSONObject channel  =
reader.getJSONObject("channel");
                    last_entry_id = channel.getInt("last_entry_id");

                    // Getting JSON Array node
                    JSONArray feeds = reader.getJSONArray("feeds");
                    feeds_length = feeds.length();
```

```
                    feeds_start = feeds_length - 24;
                    if (feeds_start < 0) feeds_start = 0;
                    myFeeds.clear();
                    // looping through All Feeds
                    for (int i = 0; i < feeds.length(); i++) {
                        JSONObject feed = feeds.getJSONObject(i);
                        int id = feed.getInt("entry_id");
                        double out_temp = feed.getDouble("field1");
                        double out_humid = feed.getDouble("field2");
                        double in_temp = feed.getDouble("field4");
                        double in_humid = feed.getDouble("field5");
                        String entry_time = feed.getString("field7");

                        myFeeds.add(new MyFeeds(id, out_temp,
out_humid, in_temp, in_humid, entry_time));

                    }
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }

            HumidChart.setDrawGridBackground(false);
            // no description text
            HumidChart.getDescription().setEnabled(false);
            // set an alternative background color
            HumidChart.setBackgroundColor(Color.WHITE);
            HumidChart.setViewPortOffsets(0f, 0f, 0f, 0f);
            HumidChart.getXAxis().setSpaceMin(1);
            HumidChart.getXAxis().setSpaceMax(1f);

            setData(feeds_length, check_place, HumidChart);

            if (check_place.equals("in")){
                HumidLabel.setText("INDOOR HUMIDITY CHART");
            }else if (check_place.equals("out")){
                HumidLabel.setText("OUTDOOR HUMIDITY CHART");
            }
            minHumid.setText(getMinHumid(feeds_length, check_place));
            maxHumid.setText(getMaxHumid(feeds_length, check_place));
        }
    }

    private void setData(int count, String place, LineChart
HumidChart) {

        ArrayList<Entry> yValues = new ArrayList<>();
        for (int i = feeds_start; i < count; i++) {
            MyFeeds addFeed = myFeeds.get(i);
            if (place.equals("in")) {
                yValues.add(new Entry(addFeed.getEntry_id(),
addFeed.getIn_humid()));
            }else if (place.equals("out")) {
                yValues.add(new Entry(addFeed.getEntry_id(),
addFeed.getOut_humid()));
            }
        }

        LineDataSet set1;

        // create a dataset and give it a type
```

```
            set1 = new LineDataSet(yValues, "Humidity");

            set1.setValueFormatter(new DecimalRemover(new
    DecimalFormat("###,###,###.##")));
            set1.setDrawCircles(false);
            set1.setLineWidth(1.8f);
            set1.setCircleRadius(4f);
            set1.setCircleColor(Color.BLUE);
            set1.setColor(Color.BLUE);
            set1.setFillColor(Color.BLUE);

            // create a data object with the datasets
            LineData data = new LineData();
            data.removeDataSet(data.getDataSetByLabel("Humidity",true));
            data.addDataSet(set1);
            data.setValueTextSize(9f);

            // set data
            HumidChart.setData(data);
            HumidChart.notifyDataSetChanged();
            HumidChart.getLegend().setEnabled(false);
            HumidChart.invalidate(); // refresh
        }
        private String getMaxHumid(int count, String place) {
            float maxHumid = (float) 0.0;
            for (int i = 0; i < count; i++) {
                MyFeeds addFeed = myFeeds.get(i);
                if (place.equals("in")) {
                    if (addFeed.getIn_humid() > maxHumid) {
                        maxHumid = addFeed.getIn_humid();
                    }
                }else if (place.equals("out")) {
                    if (addFeed.getOut_humid() > maxHumid) {
                        maxHumid = addFeed.getOut_humid();
                    }
                }
            }
            return Float.toString(maxHumid);
        }

        private String getMinHumid(int count, String place) {
            float minHumid = (float) 100.0;
            for (int i = 0; i < count; i++) {
                MyFeeds addFeed = myFeeds.get(i);
                if (place.equals("in")) {
                    if (addFeed.getIn_humid() < minHumid) {
                        minHumid = addFeed.getIn_humid();
                    }
                }else if (place.equals("out")) {
                    if (addFeed.getOut_humid() < minHumid) {
                        minHumid = addFeed.getOut_humid();
                    }
                }
            }
            return Float.toString(minHumid);
        }
    }
```

- Credits.java

```
package com.example.meteostation;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class Credits extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_credits);
    }
}
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. https://www.arduino.cc/
2. https://maker.pro/arduino/tutorial/how-to-choose-the-right-arduino-board-for-your-project
3. http://fritzing.org/home/
4. http://www.diyembedded.com/tutorials/nrf24l01_0/nrf24l01_tutorial_0.pdf
5. http://www.nodemcu.com/index_en.html
6. https://cdn-shop.adafruit.com/datasheets/Digital+humidity+and+temperature+sensor+AM2302.pdf
7. https://community.thingspeak.com/tutorials/arduino/send-data-to-thingspeak-with-arduino/
8. https://datasheets.maximintegrated.com/en/ds/DS3231.pdf
9. https://draeger-it.blog/arduino-lektion-17-realtimeclock-rtc-ds3231/
10. https://el.wikipedia.org/wiki/Android
11. https://en.wikipedia.org/wiki/Android_Studio
12. https://en.wikipedia.org/wiki/ESP8266
13. https://en.wikipedia.org/wiki/NodeMCU
14. https://github.com/PhilJay/MPAndroidChart
15. https://howtomechatronics.com/tutorials/arduino/arduino-wireless-communication-nrf24l01-tutorial/
16. https://lastminuteengineers.com/nrf24l01-arduino-wireless-communication/
17. https://openhomeautomation.net/weather-station-arduino/
18. https://techtutorialsx.com/2016/05/22/esp8266-connection-to-ds3231-rtc/
19. https://thekurks.net/blog/2018/2/5/wakeup-rtc-datalogger
20. https://thingspeak.com/channels/472441
21. https://thingspeak.com/pages/learn_more
22. https://www.adafruit.com/product/385
23. https://www.androidauthority.com/use-remote-web-api-within-android-app-617869/
24. https://www.hackster.io/djsb/canny-esp8266-nrf24l01-mi-light-gateway-20918d

25. https://www.ibm.com/developerworks/library/iot-nodemcu-open-why-use/index.html
26. https://www.instructables.com/id/Arduino-Wireless-Weather-Station/
27. https://www.instructables.com/id/Esay-IoT-Weather-Station-With-Multiple-Sensors/
28. https://www.mathworks.com/help/thingspeak/rest-api.html
29. https://www.sparkfun.com/datasheets/Components/nRF24L01_prelim_prod_spec_1_2.pdf
30. https://www.techsfo.com/blog/2014/01/charts-on-android-with-webview/
31. https://www.youtube.com/watch?v=MiVx3AQD_PI