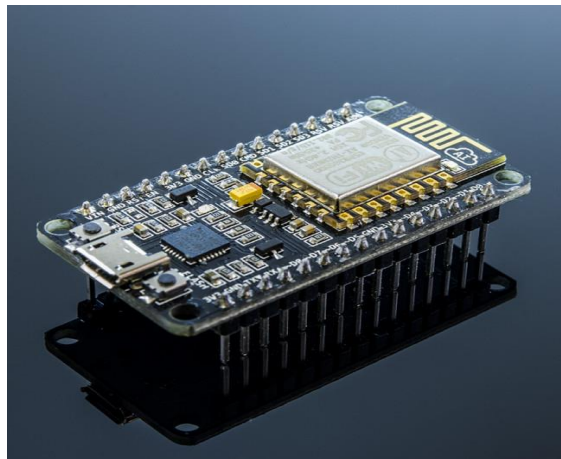


# Αυτοματοποίηση σπιτιού με χρήση αισθητήρων σε mesh network μικροελεγκτών.



Γιώργος Σταματάκης – 2013 030 154

Σπυριδάκης Χρήστος – 2014 030 022

## ΣΚΟΠΟΣ ΤΗΣ ΕΡΓΑΣΙΑΣ

Ο στόχος μας είναι να φτιάξουμε ένα δίκτυο πλέγματος (mesh network) από μερικά esp8266 (βλ. εικόνα εξωφύλλου) που είναι οικονομικά Wi-Fi modules με πολύ μικρή κατανάλωση ,κάτι που τα καθιστά ιδανικά για mesh δίκτυα. Πάνω σε αυτό το δίκτυο θα μεταφέρεται πληροφορία από διάφορους αισθητήρες που θα έχουν τοποθετηθεί πάνω στα esp8266 και θα καταλήγει σε ένα σταθμό βάσης που θα αποθηκεύει τα δεδομένα σε εξωτερική persistent μνήμη. Ανάλογα με τις μετρήσεις θα ενεργοποιούνται οι αντίστοιχες συσκευές μέσα στο χώρο (πχ. Σε περίπτωση φωτιάς θα ενεργοποιείται το buzzer). Τέλος θα γίνεται εξαγωγή στατιστικών στοιχείων από τις διάφορες μετρήσεις και θα προβάλλονται κάποια από τα αποτελέσματα σε οθόνη LCD του σταθμού βάσης.

### Γιατί mesh network?

Ο βασικός λόγος είναι ότι δίνει ελευθερία τόσο στο χρήστη όσο και στον μηχανικό που το σχεδιάζει μιας και προβλήματα όπως η αυτονομία του δικτύου και η εμβέλεια των συσκευών σχεδόν εξαλείφονται. Το δίκτυο αρχικά δεν χρειάζεται να συνδεθεί με κάποιο router ή άλλη συσκευή για να δρομολογήσει ή να συντονίσει τους κόμβους του. Επίσης η ένταξη μιας νέας συσκευής στο δίκτυο δεν απαιτεί νέες ρυθμίσεις ή αλλαγές στην υπάρχουσα υποδομή, ο χρήστης μπορεί κυριολεκτικά να αγοράσει μια καινούρια συσκευή και απλά να πατήσει το ON. Επίσης η απόσταση των αισθητήρων μπορεί να είναι αρκετά μεγάλη μιας και μπορούν να χρησιμοποιηθούν ενδιάμεσοι κόμβοι σαν repeaters.

### Scalability

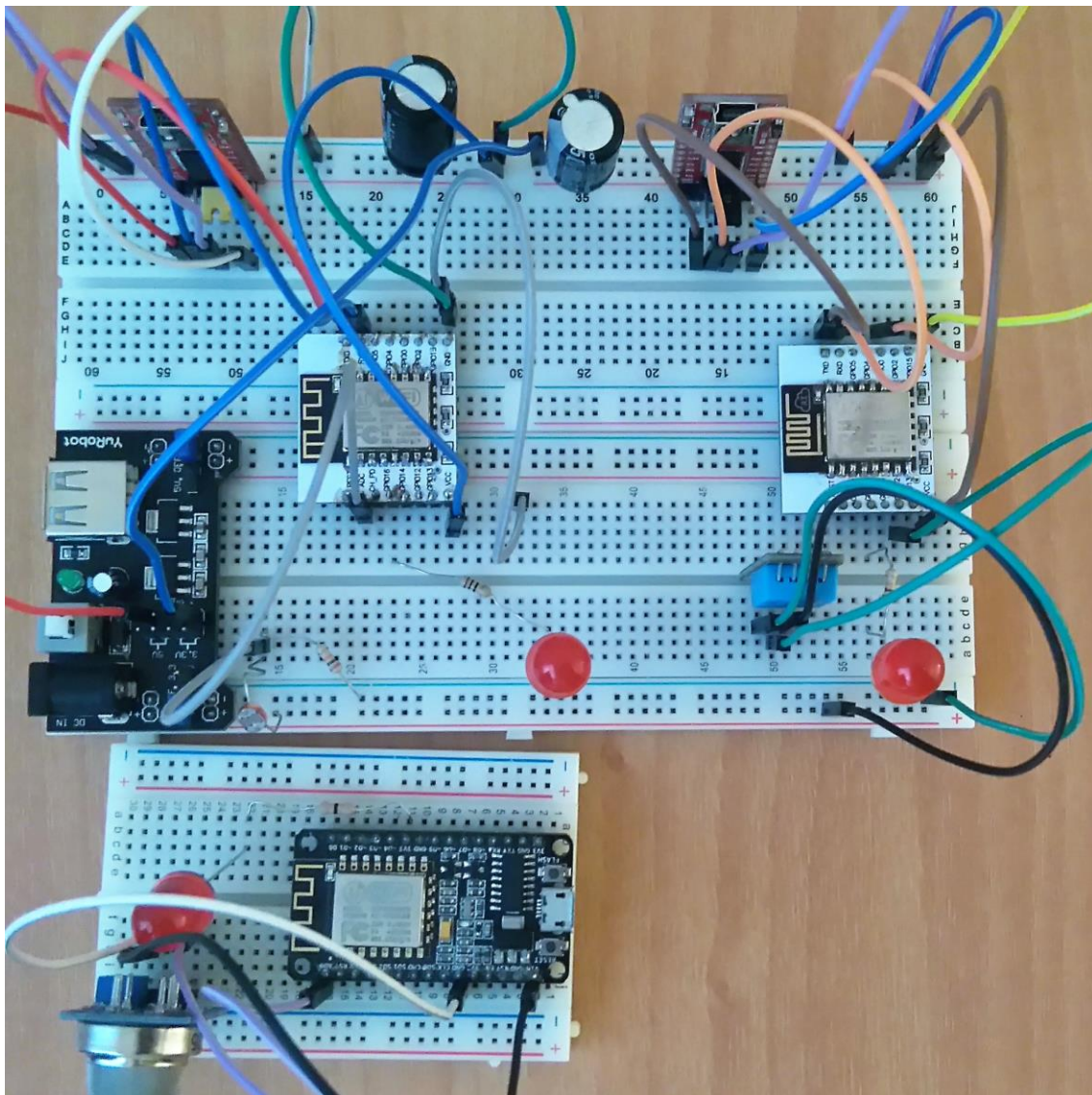
Το δίκτυο, λόγω της κατανεμημένης φύσης του, θα μπορεί να μεγαλώσει (scale out) σε πολύ μεγάλο βαθμό μιας και ο μόνος περιορισμός είναι ο αριθμός των modules που μπορεί να κρατήσει στη μνήμη (stack) του κάθε esp8266 (ένας αριθμός 16bit), ένα αρκετά μεγάλο νούμερο. Επίσης το γεγονός ότι η πληροφορία των αισθητήρων δεν περνάει μόνο από έναν κόμβο επιτρέπει μεγαλύτερους ρυθμούς μετάδοσης και καταργεί πιθανά bottlenecks.

### Γιατί esp8266?

Το συγκεκριμένο Wi-Fi module είναι συμβατό με πολλούς microcontrollers (Arduino, Raspberry, Teensy) και ταυτόχρονα έχει αρκετά καλό documentation μιας και έχει συμπληρώσει κάποια χρόνια στην αγορά. Είναι οικονομικό (κοστίζει περίπου 3\$) και καταναλώνει πολύ λίγη ενέργεια ενώ παράλληλα διαθέτει και αρκετά power saving modes (βλ. παράρτημα – πίνακα 1). Έχει αρκετά pins για τον έλεγχο όλων των αισθητήρων που σκοπεύουμε να χρησιμοποιήσουμε.

## MILESTONE 1

Στο πρώτο Milestone θέσαμε ως βασικό στόχο την εξοικείωση μας με την τεχνολογία και το υλικό που δεσμευτήκαμε να χρησιμοποιήσουμε, κάτι που πιστεύουμε ότι πετύχαμε. Αρχικά προμηθευτήκαμε όλα τα απαραίτητα υλικά για να φτιάξουμε ένα μικρό demo που αποδεικνύει ότι το mesh που δημιουργήσαμε δουλεύει γρήγορα και με αξιοπιστία. Πιο συγκεκριμένα συγκεντρώσαμε 2 esp8266-12E και άλλα 2 NodeMCU-1.0 που είναι ουσιαστικά μια αναπτυξιακή πλατφόρμα με ένα esp8266-12E πάνω της. Χρησιμοποιήσαμε επίσης 2 FTDI για τη σύνδεση των 2 esp8266-12E με υπολογιστή για λόγους τόσο τροφοδοσίας όσο και ελέγχου του serial interface των esp. Τέλος, οι 3 αισθητήρες μας (θερμοκρασίας, αερίων και φωτός) τοποθετήθηκαν σε breadboard μαζί με τα υπόλοιπα 4 esp (NodeMCU ΚΑΙ 12E ) ενώ σε αυτό το στάδιο αντί για συσκευές τοποθετήσαμε απλά LEDs. Όσον αφορά την τροφοδοσία των εξαρτημάτων τα esp8266-12E τροφοδοτούνται από FTDIs ενώ τα NodeMCU συνδέονται κατευθείαν μέσω USB, για αυτό το λόγο δεν απαιτείται FTDI τόσο για τροφοδοσία όσο και για προγραμματισμό. Να σημειωθεί επίσης ότι επειδή τα esp8266-12E απαιτούν τροφοδοσία 3.3V και ~100mA ενώ οι αισθητήρες 5V έχουμε τοποθετήσει μια μικρή συσκευή τροφοδοσίας που βγάζει 5V στην μια μεριά του Breadboard.



Όσον αφορά τον κώδικα και τις μετρήσεις η πορεία που ακολουθήσαμε ξεκίνησε βρίσκοντας μια βιβλιοθήκη που επέτρεπε σε όλα τα είδη esp8266 να δημιουργούν ένα mesh network μεταξύ τους. Επίσης, το mesh παρέχει στους κόμβους του callbacks για διάφορα σημαντικά events όπως την προσχώρηση ενός νέου κόμβου στο υπάρχον δίκτυο αλλά και διάφορες μεθόδους εύκολης επικοινωνίας τύπου signal και broadcast. Η λογική πίσω από το mesh δίκτυο είναι ότι κατά την εκκίνησή του ο μικροελεγκτής θα σηκώσει ένα ad-hoc δίκτυο με SSID που έχει ένα προκαθορισμένο prefix, στη συνέχεια θα ψάξει να βρει άλλα δίκτυα τέτοιου τύπου με το αντίστοιχο prefix για να συνδεθεί και να φτιάξει ένα mesh δίκτυο που μπορεί να μεγαλώσει αντίστοιχα. Τα μηνύματα μέσα στο mesh ανταλλάσσονται μέσα από απλές TCP συνδέσεις ενώ ο κάθε κόμβος χαρακτηρίζεται από το μοναδικό κωδικό που έχει το κάθε esp8266 πάνω του (chipId), μια πληροφορία που παρέχεται εύκολα από το API του μικροελεγκτή. Τέλος να σημειωθεί ότι ο κώδικας του mesh network (όλα τα αρχεία της μορφής easyMesh\*.\*) δεν είχε καθόλου documentation ή σχόλια κάτι που σημαίνει ότι αφιερώσαμε αρκετό χρόνο για τη συγγραφή documentation και την αφαίρεση ή αντικατάσταση κάποιων μεθόδων.

Όσον αφορά τους διάφορους αισθητήρες που έχουμε συνδέσει με τα esp8266 και τα LEDs (μελλοντικές συσκευές) η διαδικασία υλοποίησης ήταν απλή. Αρχικά βρήκαμε και χρησιμοποιήσαμε τις βιβλιοθήκες που παρέχουν οι κατασκευαστές για τους αισθητήρες τους διαβάζοντας παράλληλα τις προδιαγραφές και το documentation του κάθε ενός. Να σημειωθεί εδώ ότι όλοι οι αισθητήρες μας λειτουργούν με 5V είσοδο ενώ τα esp8266 με 3.3 αυστηρά με αποτέλεσμα να έχουμε στα 2 κολλητά breadboard μας διαφορετική τροφοδοσία στην αριστερή και δεξιά μεριά αντίστοιχα. Επίσης η βιβλιοθήκη για το DHT11 του κατασκευαστή δεν λειτουργούσε σωστά μιας και ήταν για Arduino και όχι για esp8266 για αυτό και χρησιμοποιήσαμε μια διαφορετική βιβλιοθήκη, για περισσότερες πληροφορίες δείτε στην τελευταία σελίδα όλες τις πηγές μας. Στη συνέχεια ορίσαμε κάποιες τιμές των αισθητήρων σαν ανώτερα (ή κατώτερα) κατώφλια και κάθε φορά που παραβιάζονται ενεργοποιείται μια από τις συσκευές μας. Για παράδειγμα αν η φωτοαντίσταση πέσει σε πολύ χαμηλές τιμές ( > 500 ) τότε θα ανάψει το LED που αντιστοιχεί στη συσκευή που κανονικά θα άναβε το LED strip. Υπενθυμίζουμε ότι οι ADCs των esp8266 είναι 10bit για αυτό και οι τιμές τους θα απεικονίζονται με uint8\_t αριθμούς. Τέλος ο MQ135 ανιχνεύει αρκετά αέρια (πχ. βουτάνιο, CO..) αλλά δεν μπορεί να τα ξεχωρίσει για αυτό σε περίπτωση που ανιχνεύσει βουτάνιο και CO θα παρουσιάσει ένα αθροιστικό αποτέλεσμα σε PPM, παρόλα αυτά για τη δουλειά που το θέλουμε εμείς (ανίχνευση CO και βουτανίου) κάνει εξαιρετική δουλειά αλλά θα πρέπει να «κάψει» για περίπου 1 ώρα πριν αρχίσει να βγάζει σωστές μετρήσεις.

Όσον αφορά τη διαχείριση της πληροφορίας που παράγουν οι αισθητήρες αλλά και τη μετάδοσή της στο mesh τα βήματα που ακολουθήθηκαν αναφέρονται ρητά στο αρχείο WSN.ino και το βοηθητικό header file meshConstants.h. Αρχικά στο setup phase αρχικοποιείται το mesh network με τις κατάλληλες παραμέτρους ( SSID, Password ) , οι διάφορες μεταβλητές και callbacks που θα χρησιμοποιηθούν από τα sketch μας, οι software timers που χρησιμοποιούμε αλλά και οι hardware timers που παρέχει ο μικροελεγκτής. Οι software timers (os\_timer\_t) εκτελούν περιοδικά διάφορα tasks όπως την ανάγνωση των τιμών των ελεγκτών μας αλλά και διάφορα maintenance tasks του mesh μας, οι περίοδοι των timers είναι ορισμένοι στο αρχείο meshConstants.h. Ο hardware timer (timer0) είναι υπεύθυνος για την αποστολή δεδομένων και ανά ένα συγκεκριμένο αριθμό κύκλων εκτελεί μια ISR ( Interrupt Service Routine) που αλλάζει την τιμή μιας μεταβλητής που με τη σειρά της ελέγχεται μέσα στο loop() και κάνει broadcast στο mesh τις τιμές του/των αισθητήρα/ων μας. Τέλος να σημειωθεί ότι το esp8266 διαθέτει Watchdog timer των 8s για περιπτώσεις ανάγκης ενώ σε περίπτωση που δεν κληθεί η εντολή delay() ή τελειώσει η loop() μέσα σε 1sec το software watchdog του esp8266 κάνει softlock το sketch μας.

Συνολικά το Memory Footprint μας είναι μικρό μιας και το σκέτσο μας πιάνει μόνο το 57% της μνήμης του esp8266 έτσι αν μελλοντικά θελήσει κάποιος να επεκτείνει το σχέδιο μας μπορεί να το κάνει εύκολα.

Ακολουθούν κάποια κομμάτια από τα logs των 3 esp8266 (2 αισθητήρες και 1 relay):

Αρχικά το παρακάτω κομμάτι του log είναι το πιο κλασικό παράδειγμα που βλέπει ένας παρατηρητής και παρουσιάζει 2 esp8266 να κάνουν broadcast τις μετρήσεις τους (το κομμάτι αυτό το παρατηρεί ένα 3<sup>ο</sup> esp8266 που λειτουργεί απλά σαν repeater).

```
Connection count: 2
Received from 16445372 : Gas PPM: 163
Received from 16445372 : C:24.00 H:55.00
Received from 16445372 : Gas PPM: 162
Received from 16445372 : C:24.00 H:55.00
Received from 16445372 : Gas PPM: 165
```

Ακολουθεί ένα απόσπασμα από την είσοδο ενός esp στο mesh.

```
0x8 -->scan started @ 179241403<--
0x8 stationScanCb():-- > scan finished @ 181366979 < --
0x8 found : mesh1461612, -48dBm0x8 MESH_PRE< ---0x8
0x8 found : stamatakis, -75dBm0x8
0x8 found : OTE WiFi Fon, -74dBm0x8
0x8 found : mesh16445372, -52dBm0x8 MESH_PRE< ---0x8
0x8 Found 2 nodes with _meshPrefix = "mesh"
0x8 connectToBestAP():0x8 findConnection(16445372): did not find connection
0x8 connectToBestAP(): Best AP is mesh16445372<---
0x4 stationStatus Changed to STATION_CONNECTING
0x8 manageConnections(): dropping 1461612 NODE_TIMEOUT last=180428708
node=183490327
0x8 closeConnection(): conn-chipId=1461612
0x8 meshDisconCb(): 0x8 AP connection. No new action needed. local_port=5555
0x8 wifiEventCb(): EVENT_SOFTAPMODE_STADISCONNECTED
0x8 wifiEventCb(): EVENT_STAMODE_CONNECTED ssid=mesh16445372
0x8 wifiEventCb(): EVENT_STAMODE_GOT_IP
0x8 tcpConnect(): Got local IP=192.168.188.2
0x8 tcpConnect(): Dest IP=192.168.188.1
0x8 tcpConnect(): connecting type=16, state=0, local_ip=192.168.188.2,
local_port=34443, remote_ip=192.168.188.1 remote_port=5555
...
Connection count: 2
```

Υπό κανονικές συνθήκες τα esp απλά θα μεταδίδουν τις μετρήσεις τους στο δίκτυο. Περιοδικά θα γίνεται ένα scan από κάθε συσκευή για να δει αν πρέπει να γίνει κάποιο join, σε περίπτωση που χρειάζεται τότε το esp απλά συνδέεται στο esp8266 με το καλύτερο σήμα. Η ανταλλαγή στοιχείων αυτών των 2 esp8266 γίνεται μέσα από μια direct TCP σύνδεση. Στο τέλος εκτελείται το callback νέας σύνδεσης σε κάθε κόμβο του δικτύου.

## ΠΑΡΑΡΤΗΜΑ

Πίνακας 1 – Κατανάλωση esp8266

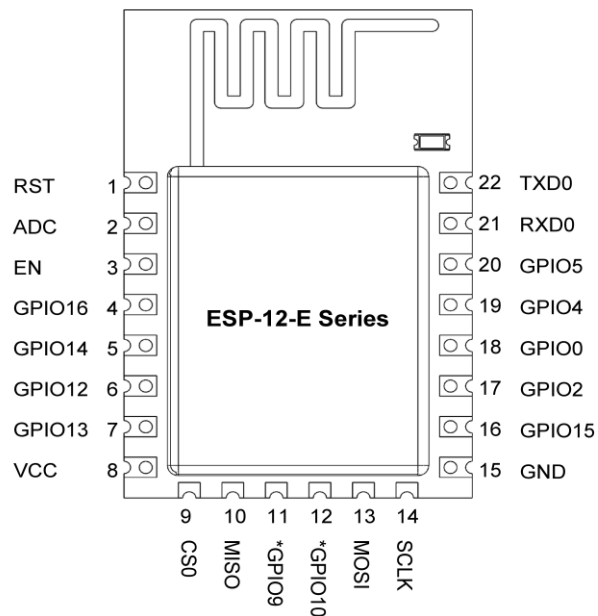
Parameter	Typical	Unit
Tx 802.11b, CCK 11Mbps, $P_{OUT}=+17\text{dBm}$	170	mA
Tx 802.11g, OFDM 54Mbps, $P_{OUT}=+15\text{dBm}$	140	mA
Tx 802.11n, MCS7, $P_{OUT}=+13\text{dBm}$	120	mA
Rx 802.11b, 1024 bytes packet length, $-80\text{dBm}$	50	mA
Rx 802.11g, 1024 bytes packet length, $-70\text{dBm}$	56	mA
Rx 802.11n, 1024 bytes packet length, $-65\text{dBm}$	56	mA
Modem-Sleep	15	mA
Light-Sleep	0.5	mA
Power save mode DTIM 1	1.2	mA
Power save mode DTIM 3	0.9	mA
Deep-Sleep	10	uA
Power OFF	0.5	uA

### Esp8266-12E pinout

ESP-12-E Series, ESP-12-D/ESP-12-Q 32Mbit

AI-THINKER

#### Pin Configuration and Functions



\* Can only be used on ESP12-D.

## ΒΙΒΛΙΟΓΡΑΦΙΑ

- <https://github.com/gstamatakis/MeshNetwork.git> Όλα τα αρχεία που χρησιμοποιήθηκαν για αυτή την εργασία (κώδικας, αναφορές, σχηματικά). Περιέχει επίσης στα Releases τα παραδοτέα για όλα τα Milestones αλλά και το τελικό προϊόν.
- <https://github.com/blackhack/ArduLibraries/tree/master/SimpleList> Χρησιμοποιήθηκε για την γρήγορη μεταφορά δεδομένων από κόμβο σε κόμβο.
- <https://github.com/bblanchon/ArduinoJson> Βιβλιοθήκη για δημιουργία και επεξεργασία JSON αρχείων σε Arduino.
- <https://github.com/gstamatakis/easyMesh.git> Βιβλιοθήκη για δίκτυα πλέγματος σε Arduino.
- <http://arduino-esp8266.readthedocs.io/en/latest/> Documentation για το esp8266.
- <http://bbs.espressif.com/viewtopic.php?t=133> esp8266 power consumption
- [https://github.com/piettetech/PietteTech\\_DHT.git](https://github.com/piettetech/PietteTech_DHT.git) DHT11 alternative library