ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΘΕΣΣΑΛΟΝΙΚΗΣ (Α.Τ.Ε.Ι.Θ.)

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ (ΣΤΕΦ)

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΑΥΤΟΜΑΤΙΣΜΟΥ Τ.Ε.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΤΙΤΛΟΣ:

"Επισκόπηση τιμών αισθητήρων σε περιβάλλον android"

TITLE:

Sensor monitoring from android devices

Γεωργιάδης Χαράλαμπος-Αβραάμ

ΑΜ: 082494

Μούκας Κωνσταντίνος

ΑΜ: 092575

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ:ΔΗΜΗΤΡΙΟΣ ΜΠΕΧΤΣΗΣ, ΚΑΘΗΓΗΤΗΣ ΕΦΑΡΜΟΓΩΝ

ΘΕΣΣΑΛΟΝΙΚΗ, ΙΟΥΝΙΟΣ 2018

ΠΕΡΙΛΗΨΗ

Σκοπός αυτής της πτυχιακής εργασίας αποτελεί η ανάπτυξη μιας εφαρμογής που θα βασίζεται σε μια διάταξη που ελέγχεται από έναν μικροελεγκτή.
Πιο συγκεκριμένα αναπτύσσεται ένα σύστημα ελέγχου μέσω του οποίου ο χρήστης θα έχει την δυνατότητα να βλέπει τη θερμοκρασία την υγρασία τη ροή υγρών, την τιμή του διοξειδίου του άνθρακα καθώς και τιμές LPG. Ιδιαίτερο χαρακτηριστικό της
εφαρμογής αποτελεί η σύνδεση του μικροελεγκτή με ένα wifi module ώστε να δίνεται η δυνατότητα στον χρήστη να έχει τον έλεγχο του συστήματος από απόσταση μέσω της αποστολής δεδομένων. Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκε το Arduino Uno R3 Atmega328p καθως και ο μικροελεγκτης ESP8266. Μία πλακέτα Arduino αποτελείται από ένα μικροελεγκτη Atmel AVR (ATmega328 και ATmega168 στις νεότερες εκδόσεις, ATmega8 στις παλαιότερες) και συμπληρωματικά εξαρτήματα για την διευκόλυνση του χρήστη στον προγραμματισμό και την ενσωμάτωσή του σε άλλα κυκλώματα.

ABSTRACT

The purpose of this dissertation is to develop an application based on a device controlled by a microcontroller. What specifically develops a control system through which o the user will be able to see the humidity temperature of the fluid flow, the carbon dioxide value and LPG values. Its special feature application is to connect the microcontroller with a wifi module to to allow the user to control the system by distance through data transmission. To implement the application, the Arduino Uno R3 Atmega328p and the ESP8266 microcontroller were used. An Arduino board consists of a Atmel AVR microcontroller (ATmega328 and ATmega168 in later versions, ATmega8 in the older ones) and complementary components to help the user to program and integrate into other circuits

ΠΕΡΙΕΧΟΜΕΝΑ

1. Εισαγωγή

Σε αυτή την ενότητα ο χρήστης εμβαθύνει στην λειτουργία του περιβάλλοντος του Arduino καθώς μαθαίνει τους κανόνες και τις απαραίτητες εντολές για την κατανόηση και συγγραφή κώδικα.

1.1. Εισαγωγή στον Arduino

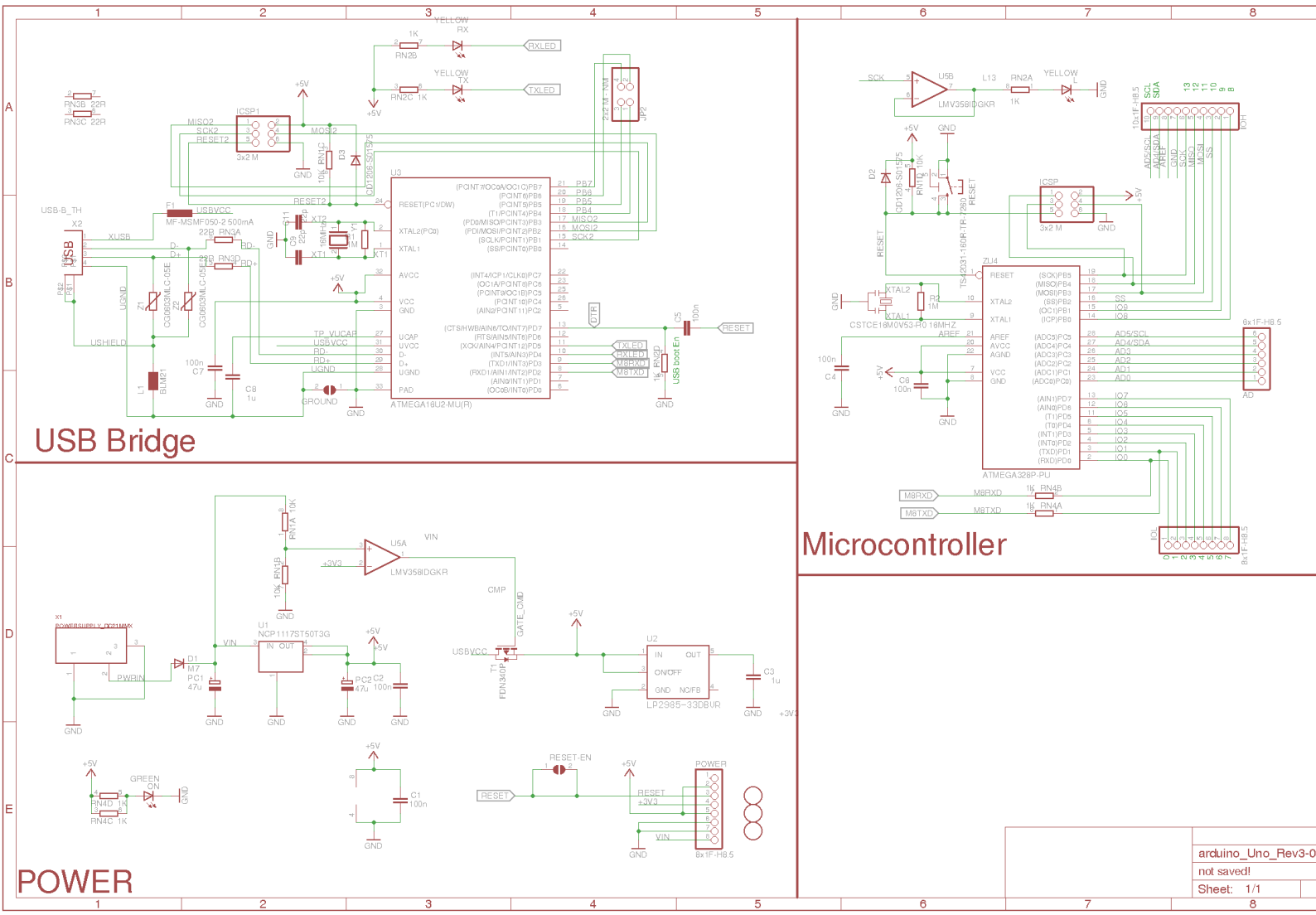
Η λέξη Arduino αναφέρεται σε μια πλατφόρμα ανοιχτού λογισμικού και υλικού με στόχο την δημιουργία διαδραστικών εφαρμογών χρησιμοποιώντας μια απλοποιημένη γλώσσα προγραμματισμού για την συγγραφή εντολών και έναν δυνατό μα απλό επεξεργαστή ο οποίος είναι σε θέση να εκπληρώσει ένα μεγάλο φάσμα καθηκόντων ανεξαρτήτως δυσκολίας[1].

Ένας ελεγκτής Arduino έχει την δυνατότητα να διαβάσει μια είσοδο, για παράδειγμα, το φως μιας λυχνίας ή την ένδειξη ενός κουμπιού και αντίστοιχα

να δώσει μια εντολή εξόδου, με την οποία να ανοίγει μια πόρτα ή να ενεργοποιείται ένας κινητήρας. Τα βασικά στοιχεία της εργαλειοθήκης του Arduino είναι ο μικροελεγκτής (microcontroller), η γλώσσα προγραμματισμού και το ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Enviroment, IDE).

1.2. Ελεγκτής Arduino (Arduino Board)

Ο μικροελεγκτής είναι ένας υπολογιστής μικρότερης κλίμακας ο οποίος χρησιμοποιεί έναν επεξεργαστή, ακροδέκτες Εισόδου/Εξόδου (I/O) και την υποδοχή USB για επικοινωνία με άλλες συσκευές και περιφερειακά καθώς και ένα μικρό ποσοστό Ram (Random-Access Memory). Σε αντίθεση με έναν κανονικό υπολογιστή η αρχιτεκτονική διαφέρει όσο αναφορά την τροφοδοσία του ελεγκτή και των συσκευών ή αισθητηρίων που συνδέονται με αυτόν[2].

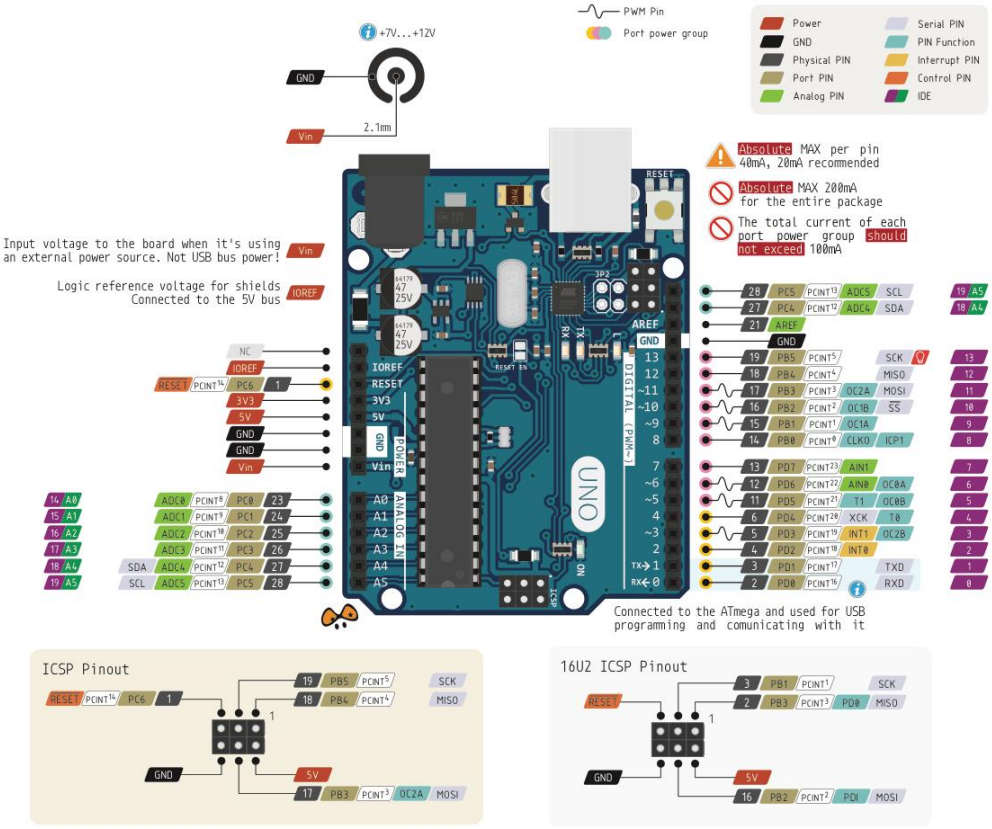


Σχεδιάγραμμα 1. Σχηματικές παραστάσεις του μικροελεγκτή, της γέφυρας usb και της τροφοδοσίας (Πηγή allaboutcircuit.com)

1.3 Περιγραφή των Στοιχείων της Πλακέτας Arduino

Στα παρακάτω δύο σχεδιαγράμματα διακρίνονται η σχεδίαση και οι ακροδέκτες

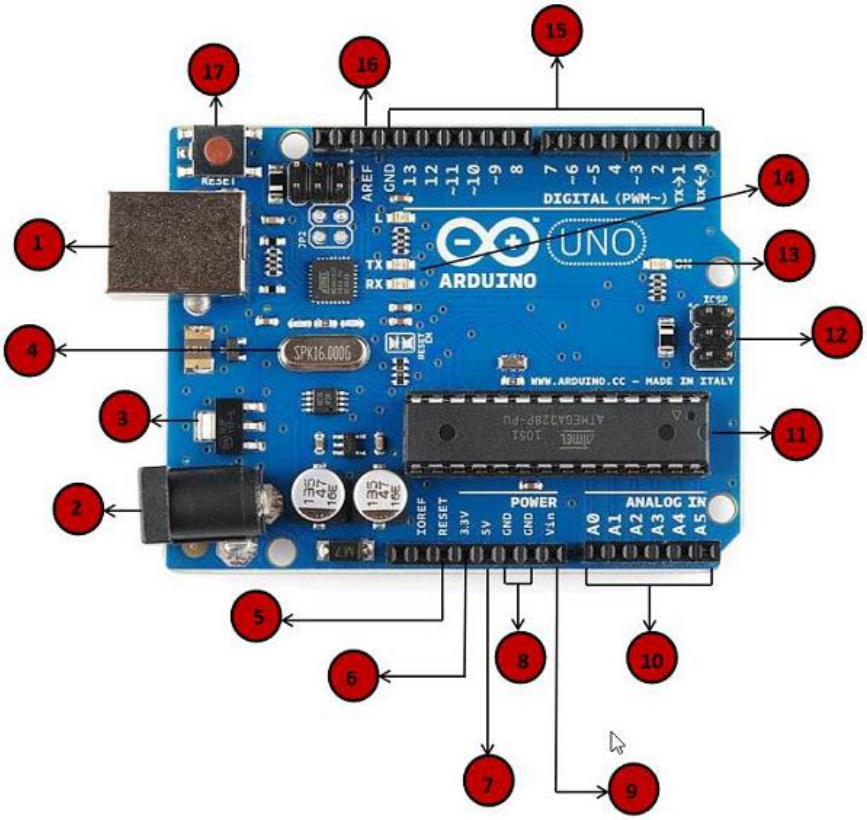
Arduino. Στην συνέχεια γίνεται περιγραφή των δομικών στοιχείων του ελεγκτή.



Σχεδιάγραμμα 2. Ακροδέκτες του Arduino Uno

2.Δομικά Στοιχεία του Arduino (Arduino Components)

Σε αυτό το κεφάλαιο δίνεται η περιγραφή των δομικών στοιχείων του Arduino Uno, λόγω της μεγάλης δημοτικότητας του ελεγκτή. Μερικοί ελεγκτές της οικογένειας του Arduino είναι διαφορετικοί από τον υποφαινόμενο όμως στην πλειοψηφία τους βρίσκει κανείς τα παρακάτω κοινά δομικά στοιχεία (βλ.



Σχεδιάγραμμα 3. Arduino Uno

Τροφοδοσία – Usb (Power - USB)

Ο ελεγκτής μπορεί να τροφοδοτηθεί από οποιαδήποτε θύρα USB ενός υπολογιστή μέσω του καλωδίου USB (1).

Τροφοδοσία (Power)

Εναλλακτικά ένας ελεγκτής μπορεί να τροφοδοτηθεί απευθείας συνδέοντας ένα τροφοδοτικό με μια πηγή εναλλασσόμενου ρεύματος. Η προτεινόμενη τάση κυμαίνεται από 7 έως 12 Volts. Σε περίπτωση που ο ελεγκτής δεχτεί τάση της τάξης των 20 Volts κινδυνεύει να καταστραφεί (2).

Ρυθμιστής Τάσης (Voltage Regulator)

Ο ρυθμιστής τάσης ελέγχει την τάση που δέχεται ο Arduino, σταθεροποιώντας την DC τάση που χρησιμοποιείται από τον μικροελεγκτή και τα υπόλοιπα στοιχεία (3).

Κρυσταλλικός Ταλαντωτής (Crystal Oscillator)

Ο κρυσταλλικός ταλαντωτής χρησιμοποιείται για την αντιμετώπιση χρονικών προβλημάτων και ανάλογα με την διατομή του παράγει και αντίστοιχη συχνότητα. Ο ελεγκτής χρησιμοποιεί τον κρυσταλλικό ταλαντωτή για να υπολογίζει χρονικά διαστήματα που παρεμβάλλονται κατά την εκτέλεση ενός προγράμματος. Η τιμή που αναγράφεται επάνω στον κρυσταλλικό ταλαντωτή είναι η αντίστοιχη συχνότητα του σε Hertz (4).

Επαναφορά (Reset)

Ο Arduino έχει την δυνατότητα να επαναφέρει το πρόγραμμα που εκτελεί την παρούσα χρονική στιγμή στην αρχική του κατάσταση χρησιμοποιώντας το κουμπί Reset, είτε συνδέοντας ένα εξωτερικό κουμπί στον ακροδέκτη Reset (5).

Ακροδέκτες

Οι ακροδέκτες είναι υποδοχές οι οποίες έχουν την δυνατότητα να συνδεθούν με μικρά μεταλλικά ή χάλκινα καλώδια, παρέχοντας έναν δίαυλο επικοινωνίας μεταξύ του ελεγκτή και ενός αισθητηρίου είτε πρόκειται για είσοδο, είτε για έξοδο. Οι ακροδέκτες με τις ενδείξεις 3.3, 5, GND, VIN χρησιμοποιούνται κυρίως για την τροφοδότηση περιφερειακών συσκευών και αισθητηρίων, αλλά και για την τροφοδότηση του ελεγκτή.

3.3 Volts – παροχή 3.3 Volt (6)

5 Volts – παροχή 5 Volt (7)

GND, Ground – Γείωση (8)

VIN – Χρησιμοποιείται για την τροφοδότηση του Arduino από εξωτερική πηγή τροφοδοσίας, για παράδειγμα μια μπαταρία των 9 Volt (9).

Αναλογικοί ακροδέκτες (Analog Pins)

Οι αναλογικοί ακροδέκτες δίνουν την δυνατότητα να γραφτούν (write) ή διαβαστούν (read) τιμές από μία ή σε μία περιφερειακή συσκευή μέσα από ένα ευρύ φάσμα πληροφοριών. Συγκεκριμένα μπορούν να γραφτούν τιμές από 0

έως 255, δηλαδή 256 διαφορετικές καταστάσεις πληροφορίας αντιστοιχιζόμενες στο εύρος της τάσης του ελεγκτή από 0 Volt έως 5 Volt, ενώ μπορούν να διαβαστούν τιμές από 0 έως 1023, δηλαδή 1024 καταστάσεις πληροφορίας και αυτές αντιστοιχιζόμενες στο ίδιο εύρος των 5 Volt. Για παράδειγμα οι τιμές αυτές μπορούν να συλλεχτούν για την θέση ενός αντικειμένου από έναν υπέρυθρο αισθητήρα ή για την φωτεινότητα μια λυχνίας LED (10). (βλ. Σχεδιάγραμμα 3).

Μικροελεγκτής (Microcontroller)

Ο μικροελεγκτής είναι ο εγκέφαλος του Arduino, είναι υπεύθυνος για την εκτέλεση των εντολών του εκάστοτε προγράμματος που φορτώνεται στον ελεγκτή. Το ολοκληρωμένο κύκλωμα διαφέρει από ελεγκτή σε ελεγκτή συνήθως όμως προέρχονται από την Atmel Company. Στην περίπτωση του Arduino Uno χρησιμοποιείται ο Atmega 328. Πρόκειται για μια συσκευή των 8-bit η αρχιτεκτονική της οποίας της επιτρέπει να χειρίζεται 8 παράλληλα σήματα δεδομένων[3] (11).

Ακροδέκτης ICSP (ICSP Pin)

ICSP (In Circuit Serial Programming) ή SPI (Serial Peripheral Interface) είναι ένα πρωτόκολλο επικοινωνίας το οποίο χρησιμοποιείται για την σύνδεση με μία ή περισσότερες περιφερειακές συσκευές, συνήθως μικροελεγκτές. Σε μια

[3]

σύνδεση SPI μία συσκευή είναι ο Κυρίαρχος (Master) και η άλλη είναι ο Σκλάβος (Slave). Κατά κανόνα υπάρχουν γραμμές επικοινωνίας μεταξύ των συσκευών:

MISO (Master In Slave Out) - Η γραμμή από την οποία ο Slave στέλνει δεδομένα στον Master MOSI (Master Out Slave In) – Η γραμμή από την οποία ο Master στέλνει δεδομένα στις περιφερειακές συσκευές.

SCK (Serial Clock) – Η γραμμή από την οποία γίνεται η εκπομπή της συχνότητας που καθορίζει τον ρυθμό αποστολής δεδομένων από τον Master προς τις υπόλοιπες περιφερειακές συσκευές (12).

Σχεδιάγραμμα 4. ICSP (In Circuit Serial Programming) (Πηγή arduino.cc)

Η λυχνία LED ανάβει όταν το Arduino είναι συνδεδεμένος με μια πηγή τροφοδοσίας. Εάν η λυχνία είναι σβηστή ενδεχομένως να υπάρχει πρόβλημα με την τροφοδοσία της συσκευής (13).

Λυχνίες LED Tx και Rx (Tx and Rx LEDs)

Οι ακροδέκτες Tx (transmit) και Rx (receive) βρίσκονται στους ψηφιακούς ακροδέκτες 1 και 0 αντίστοιχα και χρησιμεύουν για σειριακή επικοινωνία με

άλλες συσκευές. Οι αντίστοιχες λυχνίες ανάλογα με την ταχύτητα που αναβοσβήνουν, δείχνουν τον ρυθμό μετάδοσης κατά την διαδικασία αποστολής ή παραλαβής δεδομένων (14).

Ψηφιακοί Ακροδέκτες (Digital Pins)

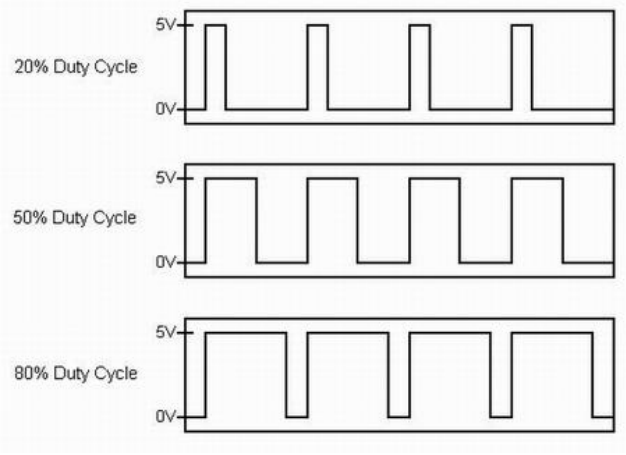
Οι ψηφιακοί ακροδέκτες έχουν την δυνατότητα να μεταβούν ενδιάμεσα σε 2 καταστάσεις: Λογικό 1 (Digital High or High) και Λογικό 0 (Digital Low or Low). Η κατάσταση High σημαίνει πως 5V στέλνονται από τον ελεγκτή στον ακροδέκτη ή διαβάζονται από μια συσκευή ή αισθητήριο, η κατάσταση Low δηλώνει πως η τάση στον ακροδέκτη είναι 0 V. Η χρήση των ψηφιακών ακροδεκτών μπορεί να φανεί ιδιαίτερα χρήσιμη στις περιπτώσεις όπου θέλει κανείς να διαχωρίσει μια λειτουργία σε 2 καταστάσεις όπως open/closed ή να ενημερώσει για την κατάσταση μιας διαδικασίας με κάποιο μήνυμα όπως για παράδειγμα "έτοιμος/όχι έτοιμος"[4] (15).

PWM (Pulse-Width Modulation)

Το Arduino χρησιμοποιεί την τεχνική PWM για να ελέγξει αναλογικά κυκλώματα με την χρήση ψηφιακής εξόδου. Ένας ψηφιακός ακροδέκτης χρησιμοποιεί την κατάσταση High (5V) και την κατάσταση Low (0V) παράγοντας έναν τετραγωνικό παλμό. Με την τεχνική PWM μπορεί κανείς να ελέγξει τον κύκλο

[4]

λειτουργίας χρησιμοποιώντας εάν ποσοστό του παλμού δίνοντας την δυνατότητα να προσομοιώσει κανείς τάσεις μεταξύ 0 και 5V. [5]



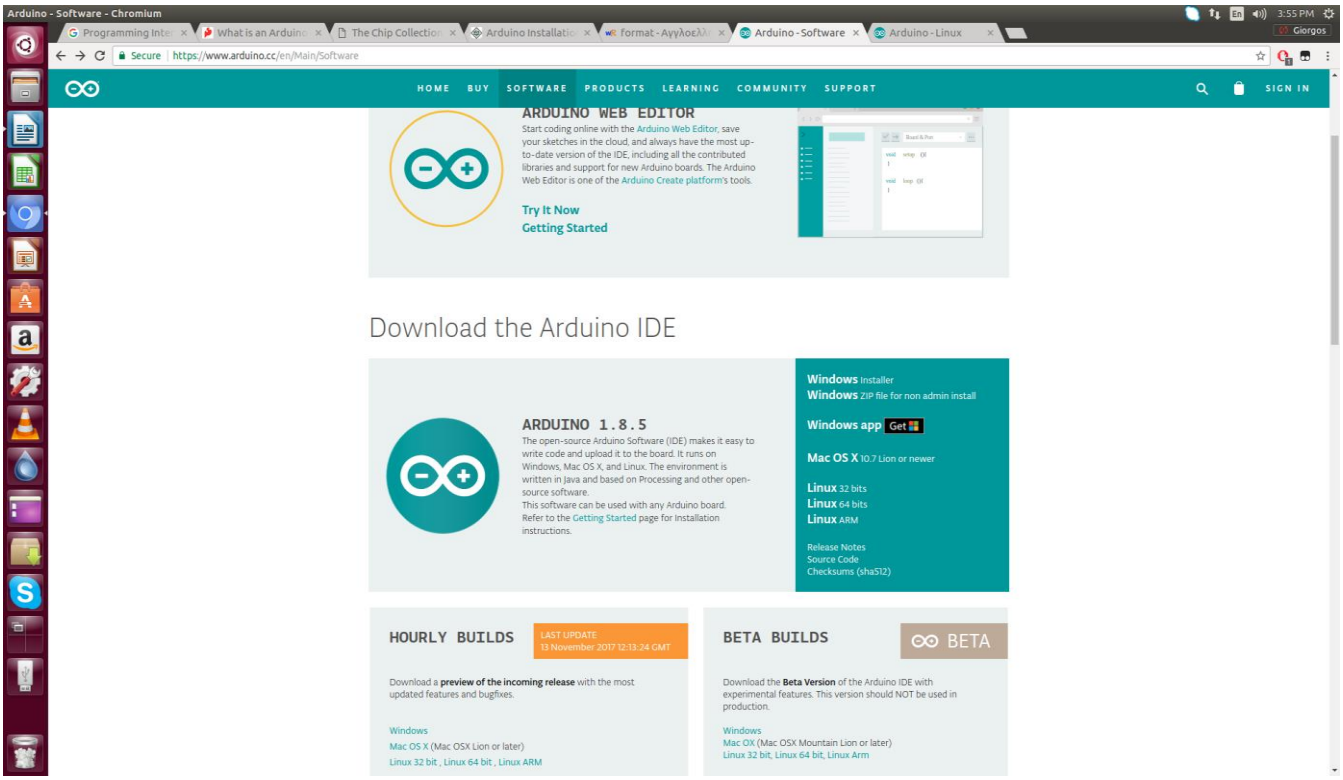
Σχεδιάγραμμα 5. PWM – Duty Cycle

Τα περισσότερα εξαρτήματα που χρησιμοποιούνται από το Arduino δουλεύουν ικανοποιητικά στις τάσεις 3.3 Volt και 5 Volt, υπάρχουν όμως και εξαιρέσεις για αυτό τον λόγο προτείνεται η ανάγνωση του φύλλου δεδομένων (data sheet) πριν την χρήση ενός εξαρτήματος. Για την παρούσα πτυχιακή χρησιμοποιήθηκε ο ελεγκτής Arduino Mega 2560 Rev 3.

2.1 Εγκατάσταση του Περιβάλλοντος Ανάπτυξης (Arduino IDE)

Το επόμενο βήμα είναι η εγκατάσταση του περιβάλλοντος ανάπτυξης (Arduino IDE), το οποίο μπορεί κανείς να το κατεβάσει από την επίσημη ιστοσελίδα του

Arduino[6] (βλ. Εικόνα 1). Ανάλογα με το λειτουργικό σύστημα που τρέχει κανείς στον υπολογιστή πρέπει να επιλέξει και την αντίστοιχη έκδοση.



Εικόνα 3. Εγκατάσταση του περιβάλλοντος ανάπτυξης

2.2 Βιβλιοθήκες του Arduino (Arduino Libraries)

Το επόμενο βήμα είναι η συμπερίληψη βιβλιοθήκης στο περιβάλλον του Arduino. Η βιβλιοθήκη είναι ένα αρχείο γραμμένο σε γλώσσα C ή C++ το οποίο δίνει σε ένα πρόγραμμα μια επιπλέον λειτουργία, για παράδειγμα τον έλεγχο ενός πίνακα LED ή τον έλεγχο στροφών ενός κινητήρα. Υπάρχουν ήδη

6

προεγκατεστημένες βιβλιοθήκες στο περιβάλλον του Arduino, αλλά δίνεται η δυνατότητα στον χρήστη να εισάγει δικές του βιβλιοθήκες.[7]

Υπάρχουν 2 τρόποι για να προσθέσει κανείς βιβλιοθήκες. Ο πρώτος τρόπος είναι να χρησιμοποιηθεί ο Διαχειριστής Βιβλιοθήκης (Library Manager) με τον εξής τρόπο:

Sketch Include Library → Manage Libraries

Εφόσον ακολουθήθηκε η παραπάνω διαδικασία, θα εμφανιστεί ο Διαχειριστής Βιβλιοθήκης (Library Manager).

Επιλέγεται η βιβλιοθήκη που επιθυμεί ο χρήστης. Εφόσον γίνει έλεγχος της πιο πρόσφατης έκδοσης της βιβλιοθήκης που πρόκειται να εγκατασταθεί, επιλέγεται το κουμπί εγκατάσταση. Η βιβλιοθήκη πλέον έχει συμπεριληφθεί στο περιβάλλον και κατά συνέπεια και στον κώδικα.

Ο δεύτερος τρόπος είναι να γίνει λήψη της βιβλιοθήκης σε μορφή φακέλου. Ο φάκελος έχει το όνομα της βιβλιοθήκης και περιέχει ένα αρχείο με κατάληξη .cpp, ένα αρχείο με κατάληξη .h και συχνά αρχεία με κατάληξη .txt, παραδείγματα και άλλα αρχεία που χρειάζεται η βιβλιοθήκη.

Επιλέγεται Sketch → Include Library Add .ZIP Library

Εφόσον βρεθεί ο φάκελος της βιβλιοθήκης, δεν έχει σημασία αν είναι αποσυμπιεσμένος ή όχι απλά πατά κανείς Open.

7

Εφόσον η διαδικασία είναι επιτυχημένη θα εμφανιστεί στην κάτω αριστερή γωνία του προγράμματος ένα μήνυμα το οποίο θα επιβεβαιώνει την συμπερίληψη της βιβλιοθήκης στο περιβάλλον του Arduino.

2.3 Επικοινωνία με τον Ελεγκτή

Στο Linux και σε άλλα λειτουργικά συστήματα τύπου Unix υπάρχουν κάποιοι σχετικοί κανόνες οι οποίοι ορίζουν την πρόσβαση ενός χρήστη ή μιας ομάδας χρηστών σε ένα φάκελο. Οι κανόνες αυτοί ονομάζονται άδειες (permissions) ή λειτουργίες φακέλων (file modes). Η εντολή chmod (change mode) χρησιμοποιείται για να ορίσει τον τρόπο με τον οποίο μπορεί κανείς να διαχειριστεί ένα φάκελο. Πληκτρολογείται η εντολή:

```
cd /dev
```

```
ls -a
```

Ο φάκελος /dev είναι ένας αποθηκευτικός χώρος για αρχεία τα οποία περιέχουν πληροφορίες για όλες τις συσκευές που χρησιμοποιούνται από το λειτουργικό σύστημα. Σκοπός είναι η αναζήτηση της σειριακής θύρας η οποία είναι συνδεδεμένη με τον μικροελεγκτή η οποία όπως προαναφέρθηκε είναι ορατή και μέσα από το περιβάλλον του Arduino. Η μορφή της εντολής που θα εκτελεστεί στο τερματικό είναι :

```
sudo chmod 777 <όνομα θύρας>
```

Ο αριθμός 777 με λίγα λόγια δίνει δικαιώματα ανάγνωσης, επεξεργασίας και εκτέλεσης ενός φακέλου στον χρήστη ή σε μια ομάδα χρηστών. Περισσότερες πληροφορίες στους παρακάτω συνδέσμους:

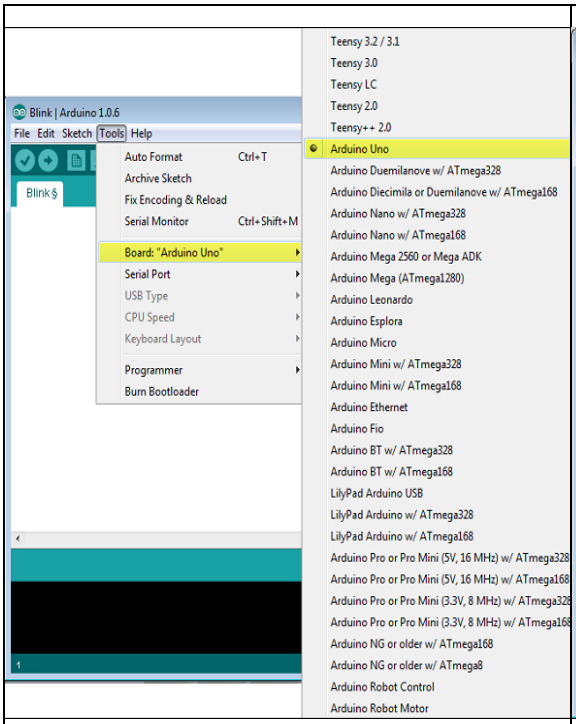
https://www.computerhope.com/unix/uchmod.htm

http://www.tldp.org/LDP/Linux-Filesystem-Hierarchy/html/dev.html

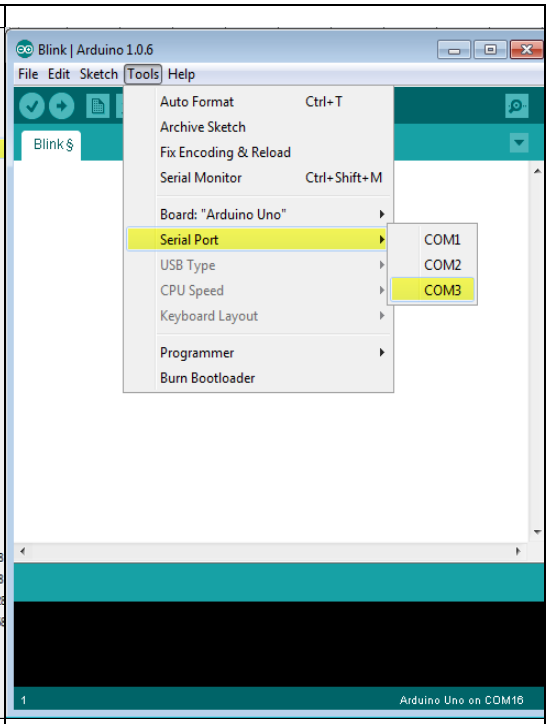
Το επόμενο βήμα είναι ο ορισμός του ελεγκτή Arduino που πρόκειται να συνδεθεί με τον υπολογιστή. Η επιλογή της πλακέτας πρέπει να είναι σωστή ώστε η ανάρτηση του κώδικα να πραγματοποιηθεί χωρίς την παρουσία σφάλματος.

Ακολουθείται η παρακάτω διαδικασία: Tools → Board (βλ. Εικόνα 8).

Τέλος γίνεται η επιλογή της θύρας με την οποία ο ελεγκτής θα συνδεθεί σειριακά με τον υπολογιστή. Για την επιλογή της θύρας επιλέγω Tools → Port (βλ. Εικόνα 9).



Εικόνα 7. Board Selection



Εικόνα 8. Port Selection

Για το ανέβασμα του προγράμματος στον ελεγκτή θα χρειαστούμε ένα καλώδιο USB A-to-B.

Πριν ξεκινήσει το ανέβασμα του προγράμματος στον ελεγκτή πρέπει να αναλυθεί η λειτουργία κάθε εικονιδίου που βρίσκεται στην γραμμή εργαλείων του Arduino IDE.

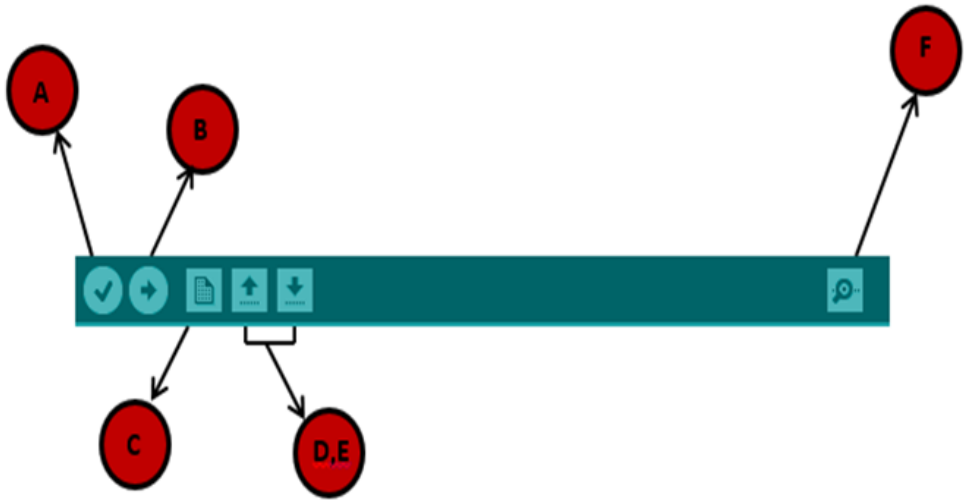
Verify (Επικύρωση)→ Γίνεται έλεγχος σφαλμάτων που μπορεί να προκύψουν στο πρόγραμμα. (A) Upload (Ανέβασμα) → Γίνεται μεταγλώττιση και ανέβασμα του προγράμματος στον ελεγκτή Arduino(B).

New (Δημιουργία) →Δημιουργία νέου προγράμματος (C)

Open (Άνοιγμα) → Άνοιγμα ενός υπάρχοντος προγράμματος (D)

Save (Αποθήκευση)→Αποθήκευση ενός προγράμματος (E)

Serial Monitor (Παρακολούθηση Σειριακής) → Εμφανίζει τα δεδομένα που στέλνονται από τον ελεγκτή στην σειριακή θύρα και τα δεδομένα που στέλνονται από την σειριακή θύρα στον ελεγκτή (F) (βλ. Εικόνα 7).



Εικόνα 7. Γραμμή εντολών της εφαρμογής Arduino (Πηγή tutorialspoint.com)

Με την επιλογή "Upload" το πρόγραμμα μεταφράζεται και ανεβαίνει στον ελεγκτή. Στον ελεγκτή οι λυχνίες Tx και Rx αρχίζουν να αναβοσβήνουν. Εάν το ανέβασμα ήταν επιτυχημένο στην κάτω δεξιά γωνία του πρόγραμμα θα εμφανιστεί η ένδειξη "Done uploading".

Σε περίπτωση που το ανέβασμα του προγράμματος στον ελεγκτή δεν είναι εφικτό είναι πολύ πιθανό η σειριακή θύρα να μην λειτουργεί σωστά . Σε αυτή την περίπτωση ο χρήστης επιλέγει Tools -> Bootloader. Σε περίπτωση που δεν δουλέψει πρέπει να γίνει είναι η αναζήτηση των "Arduino Drivers" ή η

απεγκατάσταση της ήδη υπάρχουσας εφαρμογής και εγκατάσταση της πιο πρόσφατης έκδοσης από την ιστοσελίδα του Arduino που αναγράφεται παραπάνω. Εάν και αυτή η προσέγγιση αποτύχει καλό είναι να γίνει έλεγχος του καλωδίου USB.

3. Η Γλώσσα του Arduino (Arduino Language)

Η γλώσσα που χρησιμοποιεί το Arduino έχει σχεδιαστεί ώστε να υποστηρίζει την επικοινωνία μεταξύ ηλεκτρονικών στοιχείων. Δουλεύει με παρόμοιο τρόπο όπως η γλώσσα Processing είναι όμως διαφορετικά δομημένη ώστε να αντιμετωπίζει διαφορετικά προβλήματα. Η Processing είναι μια γλώσσα ανοιχτού λογισμικού με ολοκληρωμένο περιβάλλον ανάπτυξης, σχεδιασμένη για ηλεκτρονικές τέχνες και visual design.[8]

Η ενασχόληση με τα ηλεκτρονικά στοιχεία έχει να κάνει συνήθως με την αποστολή πληροφοριών υπό μορφή τάσης ή στην περίπτωση της αρχιτεκτονικής υπολογιστών, με την αποστολή μηνυμάτων σε δυαδική μορφή. Παρόλα αυτά οι συγκεκριμένες μορφές επικοινωνίας βρίσκονται πολύ κοντά στην γλώσσα της μηχανής (Low Level). Όταν στέλνει κανείς μια εντολή σαν ένα μοτίβο ηλεκτρικών σημάτων, δουλεύει στο επίπεδο όπου τα ηλεκτρικά στοιχεία επικοινωνούν μεταξύ τους, σε σύγκριση με την αποστολή εντολής με την χρήση μιας γλώσσας υψηλού επιπέδου (High Level).

8

Η γλώσσα που χρησιμοποιεί κανείς στο προγραμματιστικό περιβάλλον του Arduino είναι βασισμένη στην γλώσσα C. Η γλώσσα C είναι μια παλιά γλώσσα προγραμματισμού, κατάλληλη στην προκειμένη περίπτωση λόγω της φύσης της κατασκευής της. Η C δημιουργήθηκε το 1972 στην επιστημονική και ερευνητική εταιρεία Bell Telephone Laboratories από τον Dennis Ritchie, σε μια εποχή όπου η υπολογιστική ισχύς ήταν δυσεύρετη.[9]

Η γλώσσα C μπορεί να μην είναι τόσο φιλική με έναν νέο χρήστη, είναι όμως οικονομική με τους πόρους της κάνοντας την ιδανική για τον προγραμματισμό μικροελεγκτών εκεί όπου οι πόροι είναι σαφώς λιγότεροι με εκείνους ενός υπολογιστή.

Εάν γνωρίζει κανείς τη γλώσσα C, θα του είναι εύκολο να μάθει να προγραμματίζει μια πλατφόρμα Arduino. Εάν κάποιος γνωρίζει τη γλώσσα Arduino, μπορεί να αντιληφθεί κάλλιστα τη γλώσσα C. Τα βασικά της γλώσσας που χρησιμοποιεί το Arduino είναι παρόμοια με την γλώσσα C++ και την Processing.[10]

3.1 Η Δομή ενός Προγράμματος

Όπως αναφέρθηκε παραπάνω η γλώσσα Arduino έχει πολλές ομοιότητες με την γλώσσα Processing όσων αναφορά την δομή του προγράμματος. Υπάρχει η συνάρτηση setup() όπου ο κώδικας μέσα στην συγκεκριμένη δήλωση τρέχει μια φορά κατά την εκκίνηση του προγράμματος και υπάρχει και η συνάρτηση loop()

η οποία εκτελείται συνεχώς. Οι περισσότερες εφαρμογές Arduino απαρτίζονται από την δήλωση μεταβλητών, οι οποίες θα χρησιμοποιηθούν κατά την διάρκεια του προγράμματος, μια αρχικοποίηση των παραμέτρων του προγράμματος ώστε να είναι έτοιμο για να λειτουργήσει και συναρτήσεις οι οποίες χρησιμοποιούνται στον κύριο βρόγχο.[11]

3.2 Η Συνάρτηση Setup (Setup Function)

Η συνάρτηση setup() είναι το πρώτο κομμάτι κώδικα που εκτελείται σε μια εφαρμογή Arduino. Έστω πως χρειάζεται η χρήση της σειριακής θύρας για την αποσφαλμάτωση ενός προγράμματος, για να εκκινήσει κανείς την σειριακή θύρα χρειάζεται να πληκτρολογήσει την συνάρτηση Serial.begin() μέσα στην συνάρτηση setup(), ώστε να επιτευχθεί η σύνδεση μεταξύ του ελεγκτή και του υπολογιστή. Η σειριακή θύρα είναι μια διεπαφή επικοινωνίας μέσω του υπολογιστή και μιας περιφερειακής συσκευής μέσα από την οποία μεταφέρονται δεδομένα. Μέσω του Serial Monitor στο περιβάλλον του Arduino ο χρήστης με το κατάλληλο μοτίβο εντολών έχει την δυνατότητα να παρακολουθεί τα δεδομένα που μεταφέρονται.

Κάποιες συσκευές πρέπει να αρχικοποιούνται κατά την εκκίνηση του μικροελεγκτή, σε άλλες συσκευές χρειάζεται η αποστολή σήματος να ξεκινήσει μετά την έναρξη τους αλλά πριν ξεκινήσει η λειτουργία τους. Όλες οι εφαρμογές πρέπει να έχουν μια συνάρτηση setup() ακόμη και στην περίπτωση που δεν έχει

11

γραφτεί τίποτα σε αυτή. Η συγκεκριμένη συνθήκη είναι απαραίτητη για τον μεταφραστή ο οποίος εμφανίζει σφάλμα σε περίπτωση απουσίας της συνάρτησης setup().[12]

3.3 Η Συνάρτηση Loop (Loop Function)

Η συνάρτηση loop περιέχει οτιδήποτε χρειάζεται να συμβαίνει συνεχώς στην εφαρμογή, για παράδειγμα, τον έλεγχο της τιμής μιας μεταβλητής, την αποστολή πληροφορίας σε έναν υπολογιστή, την αποστολή σήματος σε έναν ακροδέκτη ή τον έλεγχο της θέσης ενός ρομπότ μέσω της ανάγνωσης τιμών από έναν κωδικοποιητή. Οποιαδήποτε εντολή σε αυτή την συνάρτηση εκτελείται συνεχώς μέχρι την απενεργοποίηση της εφαρμογής.[13]

3.4 Τύποι Δεδομένων (Data Types)

Εν αντιθέσει με τον άνθρωπο, ο υπολογιστής δεν γνωρίζει την διαφορά μεταξύ "1234" και "abcd". Ο τύπος δεδομένων (data type) είναι μια κατηγοριοποίηση που προσδιορίζει τον τύπο δεδομένων που χρησιμοποιεί μια μεταβλητή και ποιες σχετικές πράξεις μπορούν να εφαρμοστούν σε αυτή χωρίς την πρόκληση σφάλματος. Για παράδειγμα ο τύπος δεδομένων string χρησιμοποιείται για την κατηγοριοποίηση αλφαριθμητικών χαρακτήρων και ένας τύπος δεδομένων int (ακέραιος) για την κατηγοριοποίηση ολόκληρων αριθμών.

12

Ο τύπος δεδομένων καθορίζει ποιες πράξεις μπορούν να κάνουν χρήση μιας μεταβλητής σε έναν υπολογισμό. Όταν μια γλώσσα προγραμματισμού απαιτεί από μια μεταβλητή να χρησιμοποιείται σε εφαρμογές που λαμβάνουν υπόψιν τον τύπο δεδομένων, αυτή η γλώσσα αναφέρεται ως αυστηρά δακτυλογραφημένη (strongly typed). Με αυτόν τον τρόπο αποφεύγονται σφάλματα, επειδή ενώ είναι λογικό να ζητήσει κανείς από τον υπολογιστή ή μικροελεγκτή να πολλαπλασιάσει έναν int με έναν float είναι παράλογο να ζητήσει κανείς πολλαπλασιασμό μεταξύ 2 μεταβλητών float και string. Όταν μια γλώσσα επιτρέπει σε μια μεταβλητή ενός τύπου δεδομένων να χρησιμοποιηθεί ως τιμή ενός διαφορετικού τύπου δεδομένων, η γλώσσα αυτή ονομάζεται αδύναμα δακτυλογραφημένη (weakly typed). Παρακάτω γίνεται αναφορά στους τύπους δεδομένων της γλώσσας Arduino.[14]

3.5 Τύποι δεδομένων της γλώσσας Arduino

void

Ο τύπος void χρησιμοποιείται μονάχα στην δήλωση συναρτήσεων. Υποδεικνύει πως η συνάρτηση αναμένεται να μην επιστρέψει κάποια πληροφορία στην συνάρτηση από την οποία καλέστηκε.

[14]

Παράδειγμα:

```
void Loop () {

        //rest of the code

}
```

bool (1 byte)

Μια boolean κρατά 2 τιμές, true ή false.

Παράδειγμα:

```
bool v = false;

bool s = true;
```

char (1 byte)

Ο τύπος δεδομένων char κωδικοποιεί τιμές χαρακτήρων και γράφονται με τον εξής τρόπο:

'A' ή "ABC" όταν πρόκειται για πίνακα χαρακτήρων.

Παρόλα αυτά οι χαρακτήρες αποθηκεύονται ως αριθμοί. Αυτό σημαίνει πως είναι εφικτό να γίνουν μαθηματικές πράξεις με την χρήση χαρακτήρων στις οποίες χρησιμοποιείται η μορφή ASCII. Για παράδειγμα η πράξη B+1 θα δώσει αποτέλεσμα 67, επειδή η τιμή του B είναι 66.

Παράδειγμα:

```
char a = 'b';

char c = 58;
```

unsigned char (1 byte)

Ο τύπος unsigned char κωδικοποιεί αριθμούς από 0 έως 255.

Παράδειγμα:

```
unsigned char y = 240;
```

byte (1 byte)

Ο τύπος unsigned char κωδικοποιεί αριθμούς από 0 έως 255.

Παράδειγμα:

```
byte a = 200;
```

int (2 bytes)

Οι ακέραιοι αριθμοί είναι ο βασικός τύπος δεδομένων για αποθήκευση αριθμών.

Έχουν εμβέλεια από -32768 έως 32767.

Παράδειγμα:

```
int a = 10;
```

unsigned int (2 bytes)

Ο τύπος unsigned int αποθηκεύει θετικές τιμές από 0 έως 65535.

Παράδειγμα:

```
unsigned int b = 50;
```

long (4 bytes)

Ο τύπος δεδομένων long κωδικοποιεί τιμές από -2.147.483.648 έως 2.147.483.647.

Παράδειγμα:

```
long a = 50000;
```

unsigned long (4 bytes)

Ο τύπος δεδομένων unsigned long αποθηκεύει μόνο θετικές τιμές από 0 έως 4.294.967.295. Έχει το ίδιο εύρος με τον long απλά μετατοπίζεται από το μηδέν και μετά.

Παράδειγμα:

unsigned long a = 10000;

float (4 bytes)

Ο τύπος δεδομένων float απεικονίζει αριθμούς με δεκαδικά ψηφία. Οι αριθμοί τύπου float χρησιμοποιούνται για να προσεγγίσουν αναλογικές και συνεχής τιμές επειδή έχουν μεγαλύτερη ανάλυση από τους ακέραιους (int). Έχουν εύρος από -3.4028235E+38 (E+38 = 10^38) μέχρι 3.4028235E+38.

Παράδειγμα:

float a = 1352,5;

double (4 bytes)

Ο τύπος δεδομένων double έχει την ίδια εφαρμογή με τον τύπο float. Ενώ θα περίμενε κανείς μια μεταβλητή τύπου double να καταλαμβάνει 8 byte στο Arduino (εκτός του Due) καταλαμβάνει 4 byte. Ο λόγος είναι ότι το Arduino είναι ένας μικροελεγκτής της τάξης των 32 bit (εκτός του Due 64 bit). 15

Παράδειγμα:

double a = 45.352;

3.6 Πεδίο Σταθερών και Μεταβλητών (Constant and Variable Scope)

Οι μεταβλητές στην γλώσσα προγραμματισμού C, την οποία χρησιμοποιεί ο Arduino έχουν μια ιδιότητα η οποία ονομάζεται πεδίο. Πεδίο είναι η περιοχή του προγράμματος όπου δηλώνονται οι μεταβλητές.

Υπάρχουν 2 είδη μεταβλητών, οι καθολικές (global) και οι τοπικές (local). Καθολική ονομάζεται η μεταβλητή η οποία αναγνωρίζεται από κάθε λειτουργία του προγράμματος. Τοπική ονομάζεται η μεταβλητή η οποία είναι ορατή μόνο στην συνάρτηση στην οποία δηλώθηκε.

Στο περιβάλλον του Arduino οποιαδήποτε μεταβλητή έχει οριστεί έξω από μια συνάρτηση (setup(), loop(), κ.α) θεωρείται ως καθολική.[16]

3.7 Τελεστές

Τελεστής ονομάζεται το σύμβολο το οποίο χρησιμοποιεί ο μεταφραστής του προγράμματος για να εκτελέσει εντολές και υπολογισμούς. Για παράδειγμα μπορεί κανείς να θέσει μια τιμή με τον τελεστή =, να ελέγξει την ισότητα δυο τιμών ή τελεστών ==, να προσθέσει με τον τελεστή + και πολλά άλλα.

Υπάρχουν 3 κατηγορίες τελεστών. Οι πρώτοι είναι οι μαθηματικοί τελεστές που εκτελούν μαθηματικές πράξεις. Οι δεύτεροι είναι οι τελεστές εκχώρησης που αλλάζουν τιμή σε μια μεταβλητή. Τρίτοι είναι οι τελεστές σύγκρισης οι

οποίοι καθορίζουν εάν 2 μεταβλητές είναι ίσες, διάφορες, μεγαλύτερες από ή μικρότερες από άλλες μεταβλητές.[17]

Τελεστής	Χρήση
+, -, *, /	Πρόσθεση, Αφαίρεση, Πολλαπλασιασμός, Διαίρεση.
%	Υπόλοιπο: Επιστρέφει το υπόλοιπο μιας διαίρεσης.
=	Εκχώρηση: εκχωρεί την τιμή στα δεξία στην μεταβλητή στα αριστερά.
+=, -=, *=, /=	Μαθηματική εκχώρηση: Προσθέτει, αφαιρεί, πολλ/ζει ή διαιρεί την τιμή/μεταβλητή στα δεξιά με την τιμή στα αριστερά και θέτει το αποτέλεσμα στην τιμή/μεταβλητή στα αριστερά.
++	Προσθέτει 1 από την τιμή/μεταβλητή στα αριστερά.
--	Αφαιρεί 1 από την τιμή/μεταβλητή στα αριστερά.
==	Συγκρίνει την τιμή/μεταβλητή στα αριστερά με την τιμή/μεταβλητή στα δεξιά. Εάν είναι ίσες η συνθήκη είναι αληθής (true).
!=	Συγκρίνει την τιμή/μεταβλητή στα αριστερά με την τιμή μεταβλητή στα δεξιά. Εάν δεν είναι ίσες η συνθήκη είναι αληθής (true).

>, >=	Συγκρίνει την τιμή/μεταβλητή στα αριστερά με την τιμή/μεταβλητή στα δεξιά. Εάν η συνθήκη είναι μεγαλύτερη ή μεγαλύτερη/ίση από την τιμή στα δεξιά η συνθήκη είναι αληθής (true).
<, <=	Συγκρίνει την τιμή/μεταβλητή στα αριστερά με την τιμή/μεταβλητή στα δεξιά. Εάν η συνθήκη είναι μικρότερη ή μικρότερη/ίση από την τιμή στα δεξιά η συνθήκη είναι αληθής (true).
&&	Ελέγχει εάν η συνθήκη είναι αληθής, αριστερά και δεξιά του τελεστή. Εάν και οι 2 είναι αληθής ολόκληρη η συνθήκη είναι αληθής.
||	Ελέγχει εάν η συνθήκη είναι αληθής, αριστερά και δεξιά του τελεστή. Είτε η μία είτε η άλλη είναι αληθής ολόκληρη η συνθήκη είναι αληθής.

Πίνακας 1. Πίνακας Τελεστών.

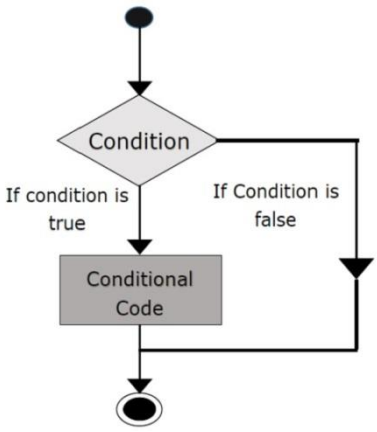
3.8 Δομές Ελέγχου και Επανάληψης (Control and Loop Statements)

Πολύ συχνά προκύπτει η ανάγκη να τροποποιήσει κανείς τις εκτελούμενες εντολές ενός προγράμματος υπό συνθήκη. Τον συγκεκριμένο σκοπό εκπληρώνουν οι συνθήκες ελέγχου και επανάληψης. Η συνθήκη ελέγχου απαιτεί τον ορισμό μιας ή περισσότερων προϋποθέσεων ώστε να ελεγχθούν από το πρόγραμμα. Εάν η προϋπόθεση είναι αληθής (true) η λογική ροή του προγράμματος θα λάβει την κατεύθυνση που έχει προκαθορίσει ο χρήστης, σε σχέση με την αντίθετη περίπτωση όπου η συνθήκη είναι ψευδής (false). Ο

βρόγχος επανάληψης εκτελεί μια λειτουργία έως ότου μια συνθήκη εκπληρωθεί ανεξάρτητα από τον αριθμό των επαναλήψεων.[18]

Δομή If/else

Η δομή if ελέγχει μια έκφραση και εκτελεί μια εντολή ή ένα πλήθος εντολών στην περίπτωση όπου η συνθήκη μέσα στην παρένθεση είναι αληθής[19].



Εικόνα 9. If Statement (Πηγή google.gr)

Παράδειγμα:

```
if (temperature >= 70)
{
  Serial.println("Danger! Shut Down the System");
}
else if (temperature >=60 && temperature < 70);
{
Serial.println("Warning! Attention required");
}
else
```

```
{
Serial.print("Safe! Continue task");
}
```

Βρόγχος Επανάληψης for (for Loop)

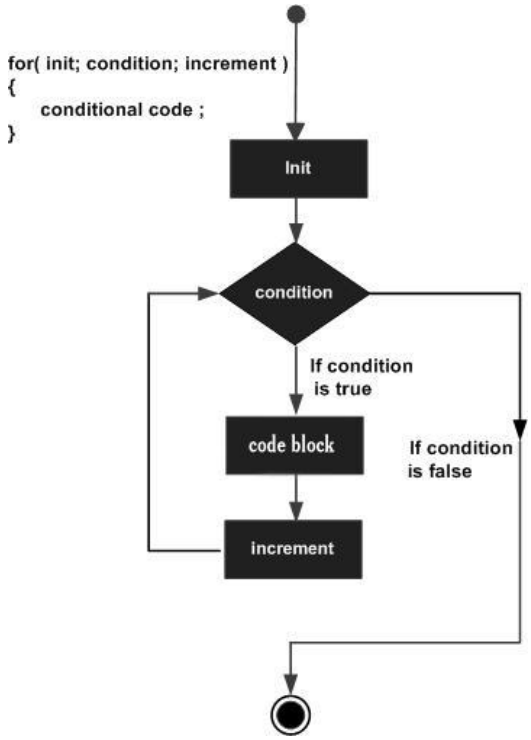
Ο βρόγχος for χρησιμοποιείται για την εκτέλεση ενός κομματιού κώδικα για συγκεκριμένο αριθμό επαναλήψεων[20].

Παράδειγμα:

```
void loop()
{
   int x = 10;
   for (int I = 0; I < x; I++)
   {
        analogWrite(13 , I);
        if( i==9)
```

{

Serial.println(“Last call”);}



Εικόνα 10. for loop

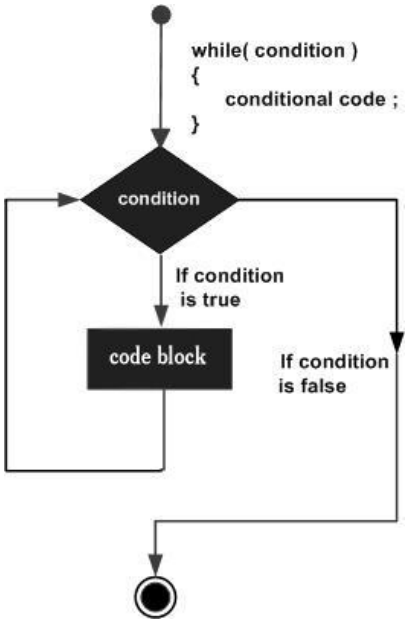
Βρόγχος Επανάληψης while (while Loop)

Η συνθήκη while επαναλαμβάνεται μέχρι η έκφραση μέσα στην παρένθεση να γινει ψευδής (false)[21].

Παράδειγμα:

```
void loop ()
{
   int var = 0;
   while (var < 200)
    {
        Serial.println(var);
```

```
        var++;    }
}
```



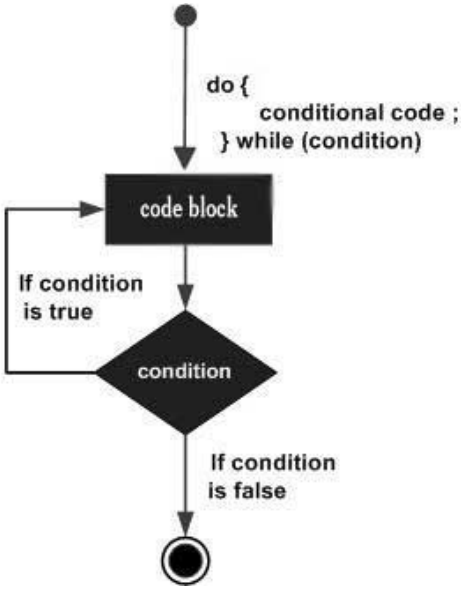
Εικόνα 11. While Loop

Βρόγχος Επανάληψης do while (Do While Loop)

Ο βρόγχος do while δουλεύει με τον ίδιο ακριβώς τρόπο όπως η while με την διαφορά πως η συνθήκη του βρόγχου ελέγχεται στο τέλος παρά στην αρχή, άρα η διαδικασία θα τρέξει τουλάχιστον μια φορά[22].

Παράδειγμα:

```
do
{
delay(50);
x= readSensors();
} while(x < 100);
```



Εικόνα 12. Do While Loop

continue

Η δήλωση continue παραλείπει την τρέχουσα επανάληψη ενός βρόγχου (For, While, Do While) και συνεχίζει προχωρώντας στην επόμενη επανάληψη[23].

Παράδειγμα:

```
for (x=0; x <= 255; x++)
{
        if (x > 40 && x < 120)
        {
                continue;
        }
```

```
        analogWrite(13 , x);
        delay(50);
}
```

break

Η δήλωση break χρησιμοποιείται για να σταματήσει έναν βρόγχο (For, While, Do While) παρακάμπτωντας την συνθήκη επανάληψης[24].

Παράδειγμα:

```
for (x=0; x < 255; x+=)
{
        analogWrite(sensor);
        if (sensor > threshold)
        {
                x = 0;
                break;
        }
        delay(50);
}
```

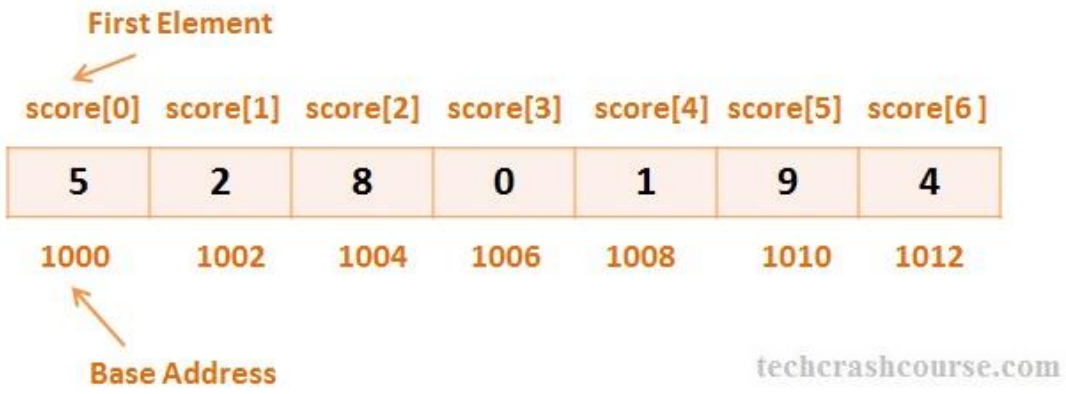
3.9 Πίνακες (Arrays)

Πίνακας είναι μια διαδοχική, από περιοχές μνήμης, ομάδα δεδομένων του ίδιου τύπου. Πιο απλά ο πίνακας είναι μια λίστα από πολλαπλά στοιχεία του ίδιου τύπου δεδομένων, για παράδειγμα int ή char. Για να αναφερθεί κανείς σε ένα στοιχείο του πίνακα πρέπει να προσδιορίσει το όνομα του πίνακα και τον αριθμό της θέσης του συγκεκριμένου στοιχείου. Η θέση ενός πίνακα ονομάζεται θέση μνήμης (base address) και η αρίθμηση των στοιχείων ξεκινά παντοτε από το 0. Έτσι λαμβάνοντας υπόψιν τον πίνακα score τα στοιχεία του προσδιορίζονται ως score[0], score[1] κ.α. Στην παρακάτω εικόνα (βλ. Εικόνα 16) διακρίνεται η θέση που καταλαμβάνει ο κάθε δείκτης στην περιοχή μνήμης[25].

Παράδειγμα:

```
int n[10];

void setup() {}

void loop()

{

  for (int I = 0;i < 10;i++)

{      n[i] = 0;

for (int j = 0;j < 10;j++)

   {

        Serial.println(n[j]);
```

}



Εικόνα 13. Array (Η base address ισχύει για όλα τα Arduino εκτός του Due και SAMD) (πηγήGoogle.gr)

3.10 Συμβολοσειρές (Strings)

Οι συμβολοσειρές μπορούν να δημιουργηθούν με δύο τρόπους, είτε με την χρήση δεδομένων τύπου String, είτε με την χρήση πινάκων τύπου char. Με τον τύπο δεδομένων String οι συμβολοσειρές δηλώνονται ως αντικείμενα. Όποιος και να είναι ο τρόπος σύνταξης μια συμβολοσειρά είναι σε θέση να αναπαριστά ακολουθίες χαρακτήρων.

Στην περίπτωση όπου μια ακολουθία χαρακτήρων απεικονιστεί με την χρήση πίνακα char πρέπει να έχει κατά νου τον μηδενικό χαρακτήρα (κώδικας ASCII

0) ο οποίος μπαίνει στο τέλος της συμβολοσειράς καταλαμβάνοντας μια θέση. Αυτό σημαίνει πως στην περίπτωση που θέλει κανείς να γράψει την λέξη "arduino" σε μορφή συμβολοσειράς η οποία απαρτίζεται από 7 χαρακτήρες, πρέπει να υπολογίσει και τον μηδενικό χαρακτήρα οπότε ο πίνακας χρειάζεται 8 θέσεις για την αναπαράσταση ολόκληρης της λέξης. Η παρουσία του μηδενικού χαρακτήρα επιτρέπει στην συνάρτηση Serial.print() να γνωρίζει πότε τελειώνει μια συμβολοσειρά[26].

Παράδειγμα:

```
void setup() {}

  char str[8];

  Serial.begin(9600);

   str[0] = 'A';

  str[1] = 'R';

  str[2] = 'D';

  str[3] = 'U';

  str[4] = 'I';

  str[5] = 'N';

  str[6] = 'O';
```

```
  str[7] = 0;
}
void loop() {
  String one = "Hello";
  String one1 = String(a);
  String one2 = String("This is a String");
  String one3 = String(13);
  String one4 = String(45 , HEX);
  String one5 = String(255 , BIN);
}
```

3.11 Συναρτήσεις (Functions)

Η συνάρτηση είναι ένα μοτίβο εντολών το οποίο είναι σε θέση να εκτελέσει μια ορισμένη λειτουργία στο σημείο του προγράμματος που κλήθηκε. Χαρακτηριστικά γνωρίσματα μιας συνάρτησης είναι το όνομα, ο τύπος δεδομένων επιστροφής, οι παράμετροι οι οποίες αντικαθιστούν τις μεταβλητές του κυρίου προγράμματος μέσα στην συνάρτηση καθώς και μια μεταβλητή/τιμή η οποία επιστρέφεται στο κυρίως πρόγραμμα και συμπίπτει με τον τύπο δεδομένων της συνάρτησης. Όπως έχει αναφερθεί παραπάνω ένα πρόγραμμα Arduino χρειάζεται τις συναρτήσεις setup() και loop() για να λειτουργήσει,

οποιαδήποτε άλλη συνάρτηση πρέπει να δημιουργηθεί έξω από τα όρια τους. Η τυπική περίπτωση για τη δημιουργία μιας συνάρτησης είναι η ανάγκη εκτέλεσης μιας συγκεκριμένης λειτουργίας αρκετές φορές σε ένα πρόγραμμα[27].

Παράδειγμα:

```
int function( int x, int y){
  int result;
  result = x * y;
  return result;
}
void loop() {
  int I = 2;
  int j = 3;
  int k;
  k = function(i , j); // Το αποτέλεσμα θα είναι 6
}
```

3.12 Συναρτήσεις Εισόδου/Εξόδου (Input/Output Functions)

pinMode (Pin , Mode)

Ένας ακροδέκτης μπορεί να οριστεί είτε σαν Είσοδος είτε σαν Έξοδος σε ένα πρόγραμμα. Η συνάρτηση pinMode() ορίζει την κατεύθυνση που θα ακολουθήσει η ροή της πληροφορίας. Η δήλωση γίνεται στην συνάρτηση setup() όπου αρχικοποιούνται μια φορά οι πληροφορίες του προγράμματος.

digitalWrite (Pin, Value)

Η συνάρτηση digitalWrite θέτει την τιμή ενός ακροδέκτη σε λογικό 1 (HIGH) ή λογικό 0 (LOW), με απλά λόγια στέλνει 5V ή 0V.

digitalRead (Pin)

Η συνάρτηση digitalRead διαβάζει την τιμή ενός ακροδέκτη επιστρέφοντας HIGH ή LOW.

analogRead (Pin)

Η συνάρτηση analogRead διαβάζει την τιμή ενός ακροδέκτη επιστρέφοντας ένα εύρος τιμών από 0 έως 1023.

delay()

Αναβάλλει την λειτουργία του πρόγραμματος για ορισμένο χρόνο σε milliseconds.

millis()

Επιστρέφει τον χρόνο λειτουργίας του προγράμματος από την έναρξη του σε milliseconds.

Κλάση (Class)

Κλάση ονομάζεται ένα πρότυπο κώδικα για την δημιουργία αντικειμένων στα οποία παρέχεται αρχικοποίηση και εφαρμογή συγκεκριμένων λειτουργιών. Το σώμα μιας κλάσης αποτελείται από τα ιδιωτικά (private) και δημόσια (public) δεδομένα. Τα ιδιωτικά μέλη της κλάσης είναι προσπελάσιμα μόνο μέσα από την κλάση και όχι από κάποια μεταβλητή που έχει δηλωθεί στο κυρίως πρόγραμμα. Τα δημόσια μέλη είναι προσπελάσιμα μέσα και έξω από την κλάση. Κάποια δεδομένα της κλάσης είναι ιδιωτικά ώστε να μην μπορούν να προσπελαστούν καταλάθος από άλλες συναρτήσεις που βρίσκονται έξω από αυτή.

Τα στοιχεία δεδομένων που περιέχονται σε μια κλάση ονομάζονται μέλη δεδομένων (data members). Οι συναρτήσεις που περιέχονται σε μια κλάση ονομάζονται συναρτήσεις-μέλη. Τα αντικείμενα της κλάσης αποτελούν την παρουσία της κλάσης μέσα στο πρόγραμμα και χρησιμοποιούν τα μέλη της ως ιδιότητές τους. Στον Arduino οι κλάσεις συνήθως περιέχονται σε αρχεία βιβλιοθηκών με κατάληξη .h και περιλαμβάνονται στον κώδικα ενός πηγαίου προγράμματος με την εντολή #include[28].

4.Μικροελεγκτής Esp8266

Τα τελευταία χρόνια αυξάνεται ραγδαία η χρήση διαφόρων μικροελεγκτών χαμηλού κόστους με μεγάλες όμως δυνατότητες. Ανάμεσα σ' αυτούς ξεχωρίζει ο μικροελεγτής NODE MCU 8266, το κόστος του οποίου μειώνεται αντιστρόφως ανάλογα της χρήσης και αποδοχής που τυγχάνει από χιλιάδες χρήστες. Αν και οι εφαρμογές που ενσωματώνουν τον NODE MCU 8266 πληθαίνουν ραγδαία, οι περισσότερες απ' αυτές ανήκουν στον χώρο της Ρομποτικής και Αυτοματισμού, της Πληροφορικής και των Ηλεκτρονικών. Στην παρούσα εργασία παρουσιάζονται τα χαρακτηριστικά και οι δυνατότητες του NODE MCU 8266 ως ένα χαμηλού κόστους σύστημα συγχρονικής λήψης και απεικόνισης με υψηλές όμως προδιαγραφές. Πιο συγκεκριμένα, η εργασία εστιάζει την προσοχή της στους αναλογικούς αισθητήρες, με τους οποίους αισθητοποιούνται δεδομένα του φυσικού περιβάλλοντος, στο σύστημα συλλογής και επεξεργασίας των δεδομένων, που το αποτελούν ο NODE MCU8266 και τα κατάλληλα ηλεκτρονικά κυκλώματα και τέλος στο τρόπο παρουσίασης των αποτελεσμάτων (smartphones,Η/Υ, LCD οθόνες,κ.λπ). Στόχος της εργασίας είναι να αναδείξει τις δυνατότητες και τα πλεονεκτήματα χρήσης του μικροελεγκτή λαμβάνοντας μετρήσεις από τα αισθητήρια και απεικονίζοντας τα σε smartphone συσκευή.

4.1 Εισαγωγή στον Esp8266

Ο nodemcu θα λέγαμε ότι είναι ένα εργαλείο για να κατασκευάσουμε ένα υπολογιστικό σύστημα με την έννοια ότι αυτό θα ελέγχει συσκευές του φυσικού κόσμου, σε αντίθεση με τον κοινό σας Ηλεκτρονικό Υπολογιστή. Είναι ανοιχτού υλικού και λογισμικού και βασίζεται σε μια αναπτυξιακή πλακέτα που ενσωματώνει επάνω έναν μικροελεγκτή και συνδέεται με τον Η/Υ για να τον προγραμματίσουμε μέσα από ένα απλό περιβάλλον ανάπτυξης. Ένας nodemcu μπορεί να χρησιμοποιηθεί για να αναπτύξουμε διαδραστικά αντικείμενα, να δεχτούμε εισόδους από πληθώρα αισθητηρίων οργάνων και διακόπτες, αλλά και να ελέγχουμε διάφορα φώτα, κινητήρες και άλλες συσκευές εξόδου του φυσικού κόσμου. Τα Projects στον εν λόγω Μικροελεγκτή μπορούν να είναι αυτόνομα (σε επίπεδο hardware) ή να επικοινωνούν με κάποιο software στον Η/Υ του προγραμματιστή.Το περιβάλλον ανάπτυξης του λογισμικού βασίζεται στην γλώσσα προγραμματισμού C++, η οποία είναι ανοιχτού κώδικα (open source) και μπορεί κάποιος να τις "κατεβάσει δωρεάν". Η Γλώσσα προγραμματισμού του nodemcu αποτελεί μια εφαρμογή σε software επίπεδο

της καλωδίωσης. Εξομοιώνει θα λέγαμε απόλυτα το φυσικό περιβάλλον του μικροελεγκτή.

4.2 Τι ειναι ο nodemcu Esp8266

Το esp8266 αποτελεί μια υπολογιστική πλατφόρμα βασιζόμενη σε μια μητρική πλακέτα ανοικτού κώδικα, που περιέχει έναν προγραμματιζόμενο μικροελεγκτή (MCU) και εισόδους/εξόδους (I/O) για τη σύνδεση με το φυσικό κόσμο. Ο μικροελεγκτής του προγραμματίζεται μέσα από το περιβάλλον ανάπτυξης κώδικα IDE χρησιμοποιώντας τη γλώσσα προγραμματισμού wiring. Η γλώσσα αυτή είναι βασισμένη στη C/C++, και περιλαμβάνει ένα σύνολο από βιβλιοθήκες υλοποιημένες στη C++. Το λογισμικό είναι ανοικτού κώδικα και επιτρέπει την ανάπτυξη ενός υπολογιστικού συστήματος το οποίο θα ελέγχει συσκευές του φυσικού κόσμου. Μπορεί επίσης να χρησιμοποιηθεί στην ανάπτυξη διαδραστικών αυτοματοποιημένων οντοτήτων, ικανών να δεχθούν ως εισόδους μια πληθώρα αισθητήριων οργάνων, αλλά και διαφόρων συσκευών εξόδου, που είναι ικανές να ελέγχουν άλλες συσκευές του φυσικού κόσμου. Γενικά μας παρέχεται η δυνατότητα να υλοποιήσουμε project αυτόνομα σε επίπεδο hardware ή αλληλοεξαρτώμενα σε επίπεδο software, που επικοινωνούν με άλλα ολοκληρωμένα μικροσυστήματα επεξεργασίας και υπολογιστές. Ειδικότερα η υπολογιστική πλατφόρμα esp8266 αποτελείται από μια πλακέτα με πυρήνα,έναν μικροελεγκτή και τον πομπό wifi.(((Διαθέτει από 8 έως 16 σειριακές θύρες και έως 50 ψηφιακές (αναλόγως τον τύπο και την έκδοση της πλακέτας) με τις οποίες αλληλοεπιδρά με διάφορες συσκευές.))

Οι αισθητήρες συνδέονται στις ψηφιακές και αναλογικές εισόδους του esp8266, ο οποίος κάνοντας χρήση των κατάλληλων βιβλιοθηκών, διαβάζει τις τιμές τάσης στην είσοδό του και τις μετατρέπει σε κατάλληλες και αναγνωρίσιμες τιμές των μετρούμενων φυσικών μεγεθών. Όλες οι πλατφόρμες esp8266 που υπάρχουν στο εμπόριο αποτελούνται από ένα γραμμικό ρυθμιστή τάσης 5V και ένα ταλαντωτή κρυστάλλου. Ο μικροελεγκτής είναι από κατασκευής του προγραμματισμένος με ένα bootloader (μικροκώδικας εκκίνησης υλικού), ώστε να μη χρειάζεται εξωτερικός προγραμματιστής όταν συνδέουμε νέο υλικό. Ο προγραμματισμός της υπολογιστικής πλατφόρμας esp8266 πραγματοποιείται μέσω USB, εφαρμόζοντας ένα chip προσαρμογέα USB toSerial.

4.3 Επιλογη του module Esp8266

Η ηλεκτρονική μονάδα ESP8266 WiFi είναι μια αυτόνομη SOC με ενσωματωμένη στοίβα πρωτοκόλλων TCP / IP που μπορεί να δώσει σε κάθε μικροελεγκτή πρόσβαση στο WiFi δίκτυο σας. Το ESP8266 μπορεί είτε να φιλοξενήσει μια εφαρμογή είτε να εκφορτώσει όλες τις λειτουργίες δικτύωσης Wi-Fi από έναν άλλο επεξεργαστή εφαρμογών. Κάθε ηλεκτρονική μονάδα ESP8266 έρχεται προ-προγραμματισμένη με ένα firmware set της συσκευής AT, που σημαίνει ότι μπορείτε απλά να το συνδέσετε στη συσκευή σας Arduino και να πάρετε όσο το δυνατόν WiFi δυνατότητες οσες το WiFi Shield προσφέρει. Η ενότητα ESP8266 είναι ένα εξαιρετικά οικονομικά αποδοτικό board με μια τεράστια και συνεχώς αυξανόμενη κοινότητα. Αυτό το module διαθέτει επαρκή ικανότητα επεξεργασίας και αποθήκευσης επί του σκάφους, η οποία επιτρέπει την ενσωμάτωσή της στους αισθητήρες και σε άλλες εφαρμογές που αφορούν συγκεκριμένες εφαρμογές μέσω των GPIO με ελάχιστη αρχική ανάπτυξη και ελάχιστη φόρτιση κατά τη διάρκεια του χρόνου εκτέλεσης. Ο υψηλός βαθμός ενσωμάτωσης σε chip επιτρέπει ελάχιστα εξωτερικά κυκλώματα, συμπεριλαμβανομένης της μονάδας μπροστινού άκρου, να είναι σχεδιασμένα να καταλαμβάνουν ελάχιστη επιφάνεια PCB. Το ESP8266 υποστηρίζει APSD για εφαρμογές VoIP και διασυνδέσεις Bluetooth, περιέχει ένα αυτο-βαθμονομημένο RF που του επιτρέπει να λειτουργεί υπό όλες τις συνθήκες λειτουργίας και δεν απαιτεί εξωτερικά εξαρτήματα RF. Υπάρχει μια σχεδόν απεριόριστη πηγη

πληροφοριών διαθέσιμη για το ESP8266, τα οποία έχουν εξασφαλιστεί από την εκπληκτική υποστήριξη της κοινότητας. Παρακάτω θα βρείτε πολλούς πόρους για να σας βοηθήσουμε να χρησιμοποιήσετε το ESP8266, ακόμη και οδηγίες για το πώς να μετατρέψετε αυτήν την ενότητα σε λύση IoT (Internet of Things)! Σημείωση: Η ηλεκτρονική μονάδα ESP8266 δεν μπορεί να μετατοπίζει λογική 5-3V και θα απαιτεί έναν εξωτερικό μετατροπέα επιπέδου λογικής. Παρακαλώ μην το τροφοδοτείτε απευθείας από το δική σας κάρτα 5V.

5 Αισθητήρες και Εξαρτήματα

5.1 Γενικά

Αισθητήρας ονομάζεται μία συσκευή που ανιχνεύει ένα φυσικό μέγεθος και παράγει από αυτό μία μετρήσιμη έξοδο. Για παράδειγμα, το υδραργυρικό θερμόμετρο μετατρέπει τη μετρούμενη θερμοκρασία σε διαστολή, η οποία μπορεί να αναγνωστεί από ένα βαθμονομημένο σωλήνα. Οι αισθητήρες χρησιμοποιούνται σε καθημερινά αντικείμενα, όπως κουμπιά ανελκυστήρων ευαίσθητα στην αφή και λάμπες φωτισμού που εκπέμπουν λαμπρότερα ή απαλότερα αγγίζοντας τη βάση τους. Υπάρχουν αναρίθμητες ακόμη χρήσεις που οι περισσότεροι άνθρωποι δεν αντιλαμβάνονται. Εφαρμογές τους συναντούμε στα αυτοκίνητα, σε μηχανές, στην αεροναυπηγική, την ιατρική, τη βιομηχανία και τη ρομποτική.

5.2 Είδη και χαρακτηριστικά αισθητήρων

Συχνά οι αισθητήρες δεν δίνουν στην έξοδο τους κατάλληλο ηλεκτρικό σήμα. τότε απαιτείται η χρήση ενος επιπρόσθετου ηλεκτρονίκου κυκλώματος, το οποίο να λαμβάνει την έξοδο του αισθητήρα και να τη μετατρέπει σε κατάλληλο ηλεκτρικό σήμα, σύμφωνα με τις απαιτήσεις των επόμενων βαθμίδων. το κύκλωμα αυτό ονομάζεται κύκλωμα ρύθμισης σήματος (signal conditioning circuit), κύκλωμα ελέγχου (control circuit) η εξωτερική μονάδα (outer η external module). για παράδειγμα, υπάρχουν αισθητήρες στάθμης που μετρούν το χρόνο που απαιτείται για να ανακλαστεί ένα υπερηχητικό κύμα απο τη μετρούμενη επιφάνεια και να επιστρέψει στο σημείο απο όπου εκπέμφθηκε. σε αυτούς πρέπει να υπάρχει κατάλληλο κύκλωμα για τη μετατροπή των τιμών χρόνου σε ανάλογες τιμές τάσης.

Οι αισθητήρες που απαιτούν εξωτερική τροφοδοσία για να λειτουργήσουν ονομάζονται ενεργοί. για παράδειγμα, ένας αισθητήρας γραμμικής μετατόπισης lvdt πρέπει να τροφοδοτειται απο καταλληλη εναλλασσόμενη τάση. οι αισθητηρες που δημιουργούν μόνοι τους μια τάση και δε χρειάζονται εξωτερική τροφοδοσία ονομάζονται παθητικοί. τέτοιοι ειναι για παράδειγμα οι πιεζοηλεκτρικοί κρυσταλλοι, που οταν πιεστουν αναπτύσσουν στα ακρα τους ηλεκτρική ταση.

Τα χαρακτηριστικά των αισθητήρων συγκροτουν τις προδιαγραφές τους (specifications) και ειναι πολλά. παρότι οι ποικιλοι αισθητήρες που υπαρχουν σημερα στηριζονται σε διαφορετικές αρχες λειτουργιας, εχουν κοινα τα βασικά τους χαρακτηριστικα. αυτα ειναι τα ακόλουθα:

1: γραμμικότητα

2: ευαισθησία

3: διακριτική ικανότητα

4: ακρίβεια

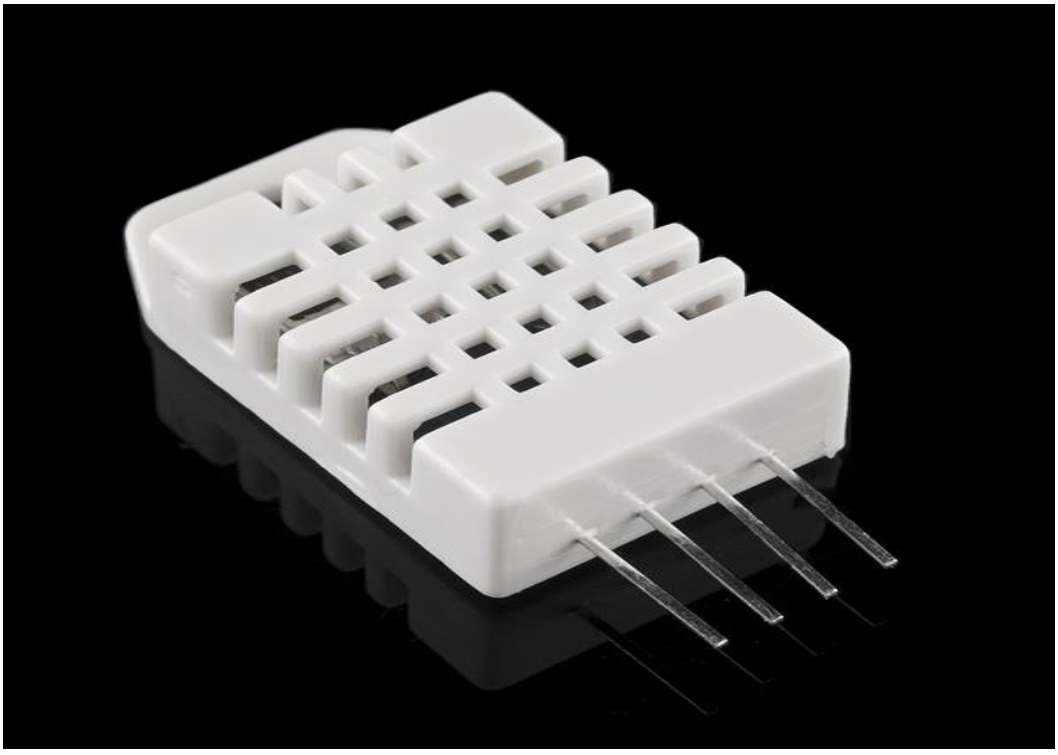
5: εύρος τιμων εισόδου

6: ευρος τιμων εξόδου

5.3 Αισθητήρας DHT-22

Το DHT-22 (γνωστό και ως AM2302) είναι ένα αισθητήριο που χρησιμοποιείτε για την εύρεση της σχετικής υγρασίας και θερμοκρασίας στον χώρο. Η υγρασία και θερμοκρασία του χώρου θα εκτυπώνονται στην σειριακή οθόνη. Το DHT-22 είναι ένας βασικός, χαμηλού κόστους, αισθητήρας για την εύρεση υγρασίας και θερμοκρασίας στον χώρο. Στο εσωτερικό του κρύβει έναν αισθητήρα υγρασίας και ένα θερμίστορ (μεταβλητή αντίσταση που η τιμή της αλλάζει σε σχέση με την θερμοκρασία) 'διαβάζοντας' έτσι τον αέρα που το περιβάλει.
Οι συνδέσεις είναι απλές, το πρώτο pin από αριστερά στα 3-5V, το δεύτερο pin (data) σε μια ψηφιακή είσοδο και το τέρμα δεξιά στην γείωση.

Τεχνικές πληροφορίες:
Πηγή : 3-5V
Μέγιστο ρεύμα: 2.5mA
Υγρασία: 0-100%, ακρίβεια 2-5%
Θερμοκρασία: -40 to 80°C, ακρίβεια ±0.5°C



Εικόνα 1 :Αισθητήρας DHT-22

5.4 Αισθητήρας αερίου MQ-2

Οι αισθητήρες MQ2 χρησιμοποιούνται για την ανίχνευση διαρροής αερίου, οικιακά αλλά και βιομηχανικά, είναι κατάλληλοι για ανίχνευση LPG βουτάνιου, προπανίου, μεθανίου, αλκοόλ, υδρογόνου και καπνού. Ίσως βρείτε και με μεταβλητή αντίσταση πάνω του η οποία ρυθμίζει την ευαισθησία του. Τα pin που διαθέτει είναι τρία: Vcc, GND και Output

Εικόνα 2 : Αισθητήρας αερίου MQ-2

5.5 Αισθητήριο μέτρησης ροής νερού

Αυτό που θα πρέπει να προσέξετε είναι η διάμετρος του (πόσα Φ είναι) ώστε να μπορείτε εύκολα να το προσαρμόσετε στην βρύση σας, η γενικότερα σε κάποια πηγή νερού (θα μπορούσε να μπει και ένα καζάνι). Το δεύτερο και σημαντικότερο είναι η πίεση που μπορεί να αντέξει (μέτρηση σε MPa) αλλά και το πόσο νερό μπορεί να περάσει μέσα απ' αυτό (μετριέται σε λίτρα ανα λεπτό L/m).

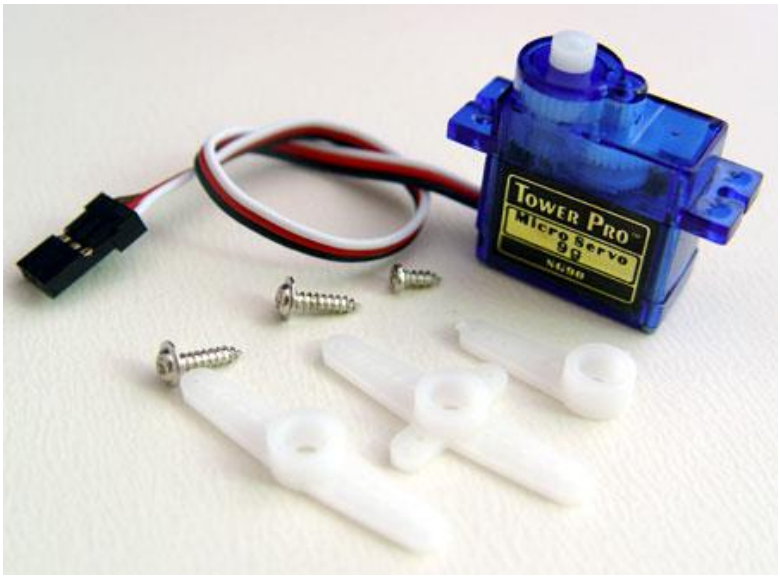
Η συνδεσμολογία είναι αρκετά απλή, το + του αισθητήρα και λογικά κόκκινο καλώδιο πάει στο Arduino pin 5V, το - του αισθητήρα και λογικά μαύρο καλώδιο πάει στο Arduino pin GND και το κίτρινο καλώδιο που είναι το σήμα μας πάει στο Arduino pin 2.

Εικόνα 3 : Αισθητήριο μέτρησης ροής νερού

5.6 Servo Micro Motor 9G SG90

Μικρό μοτέρ για Robotics & Arduino projects. Λειτουργεί από 4.2V έως 6V.

- Καλώδιο 150mm
- Βάρος: 9γρ
- Διαστάσεις: 23mm x 12.5mm x 30mm
- Ροπή 1.6kg/cm (5V)
- Ταχύτητα λειτουργίας: 0.3sec/60degree (4.8V)
- Τροφοδοσία:4.2-6V
- Θερμοκρασία: 0°-55°C
- Περιέχει 3 διαφορετικα γρανάζια και βίδες

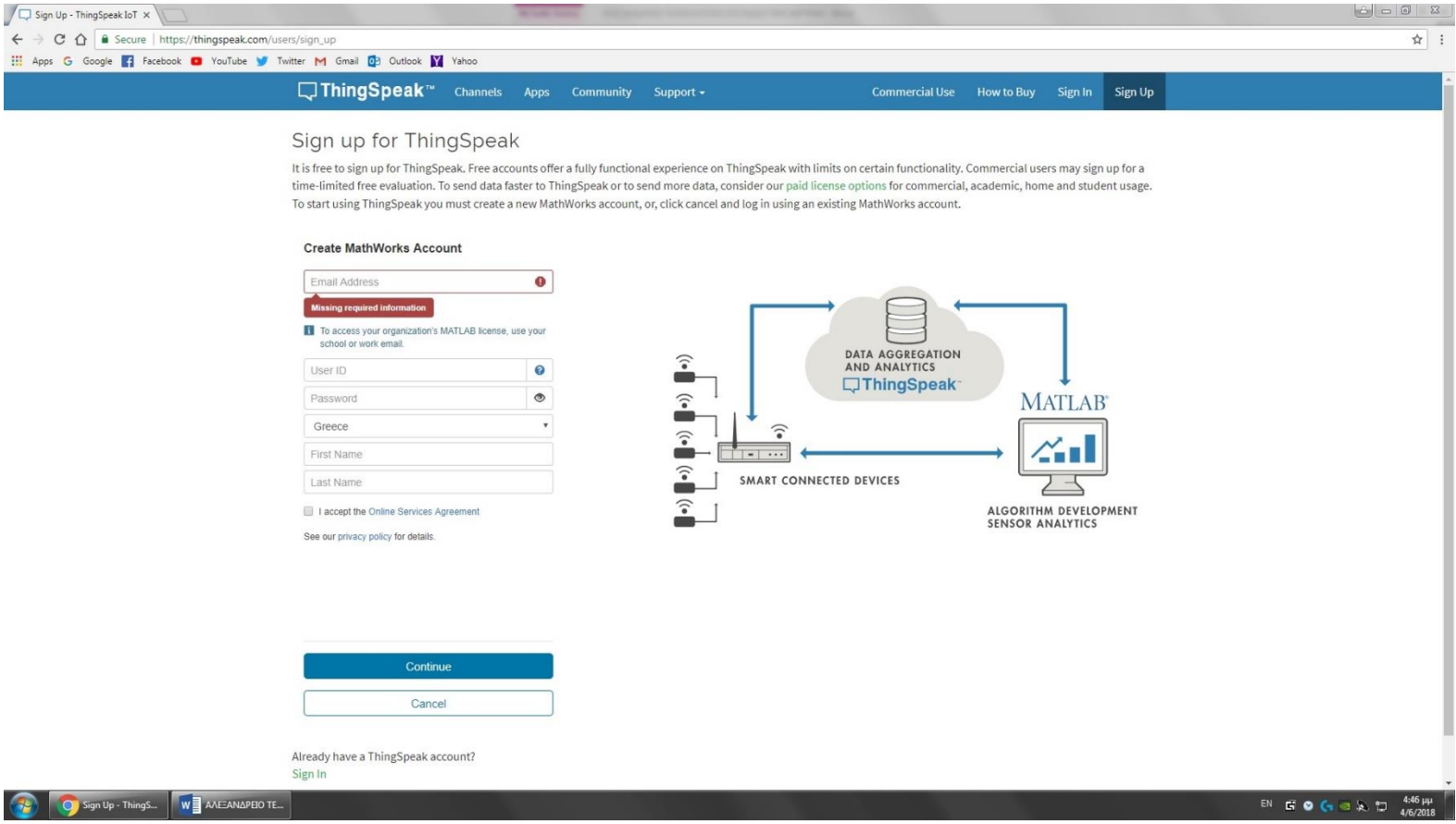


6. Πειραματικό Μέρος

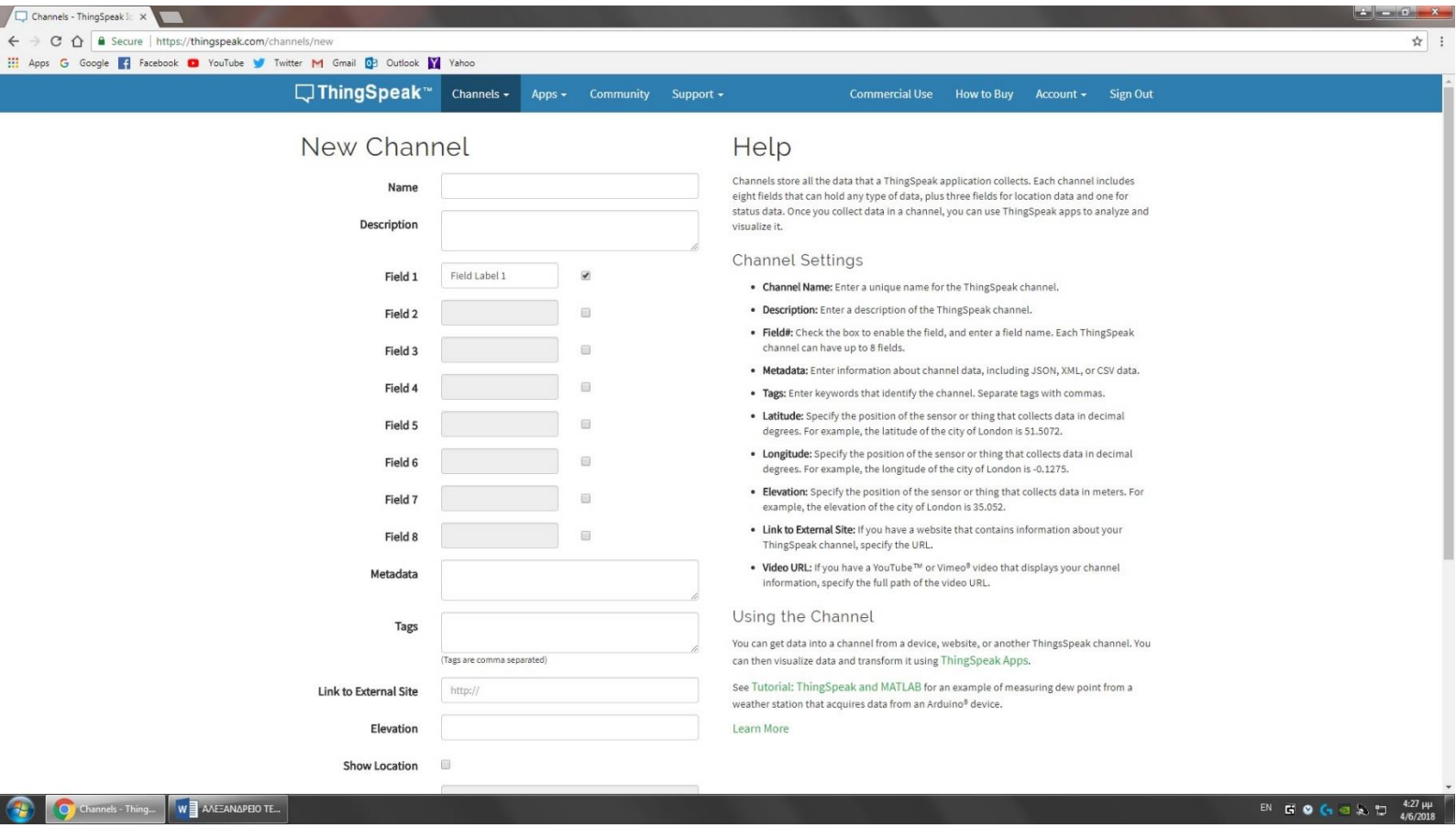
Απώτερος στόχος της όλης εφαρμογής είναι ο προγραμματισμός του Arduino, η συνδεσμολογία αισθητήρων και η μέτρηση-καταγραφή real-time τιμών θερμοκρασία, υγρασίας, καπνού, προπανίου , μεθανίου και ροης νερού του περιβάλλοντα χώρου όπου έχουμε τοποθετήσει την εφαρμογή.

6.1 Thingspeak IoT Platform

Προχωραμε στη δημιουργια ενός διαδικτυακου σερβερ από τον οποιο θα παιρνουμε τις τιμες μας και θα μπορουμε να τις βλεπουμε αναπασα στιγμη οπου και αν βρισκομαστε. Αυτό θα γινεται με την βοηθεια της σελιδας www.thingspeak.com
Μπαινουμε στην σελιδα και ξεκιναμε τη δημιουργια. Παταμε στο sign up και εκει καταχωρουμε όλα τα στοιχεια που μας ζητουνται.



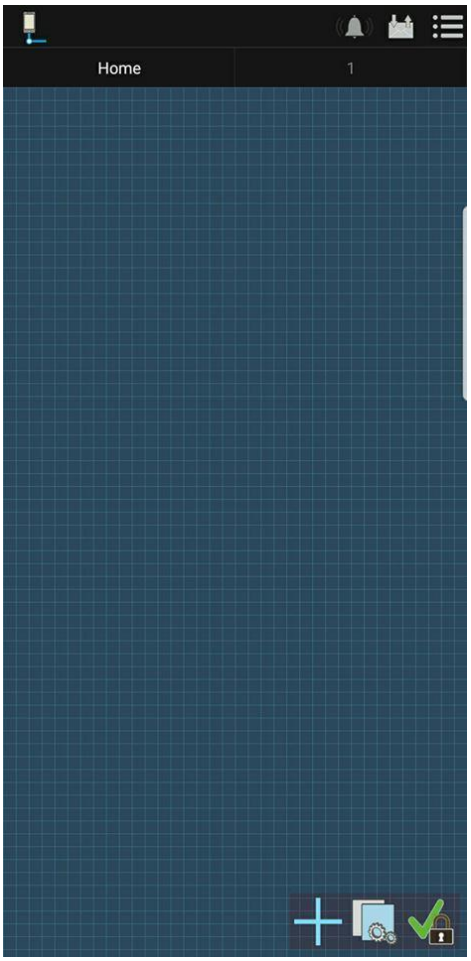
Στη συνεχεια προχωρουμε στη δημιουργια του καναλιου μας. Στα πεδια field1,field2,field3 κλπ βαζουμε τις τιμες με τη σειρα που θελουμε να μας τις εμφανιζει. Φροντιζουμε το καναλι μας να είναι σε δημοσια θεα (public view). Αυτό γινεται ενεργοποιοντας το sharing.



Αφου τελειωσουμε τη δημιουργια του σερβερ προχωραμε και στη δημιουργια της εφαρμογης η οποια θα διαβαζει τα στοιχεια που θα δεχεται από το server που δημιουργησαμε και θα μας τα εμφανιζει στο κινητο μας τηλεφωνο οπου και αν βρισκομαστε οποιαδηποτε στιγμη.

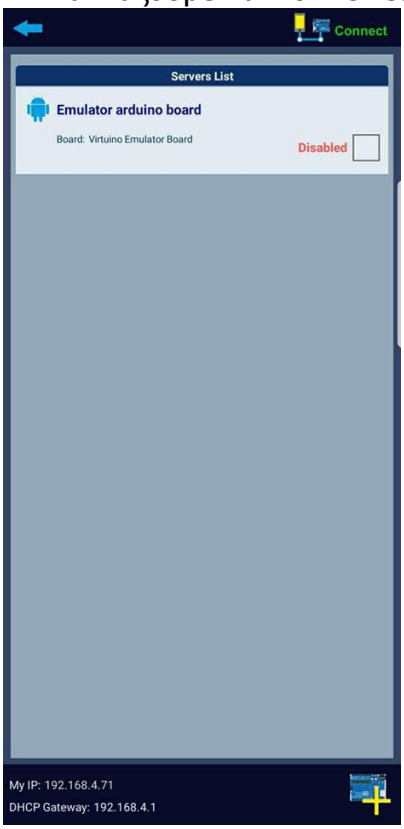
6.2 Virtuino Application

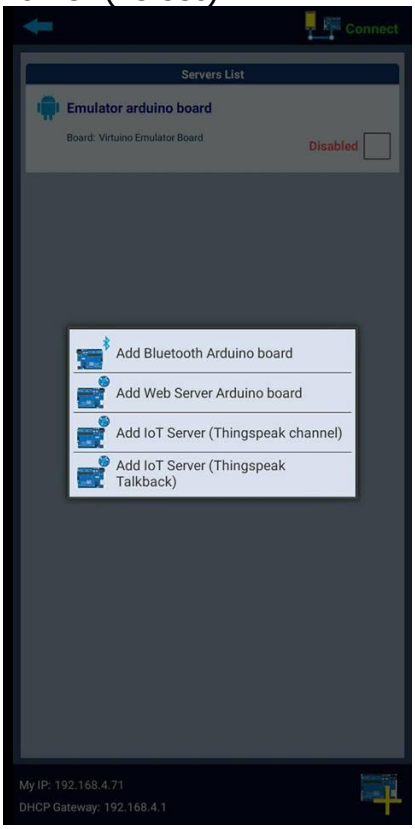
Θα χρειαστούμε μια συσκευή android για να εγκαταστήσουμε την εφαρμογή virtuino.



(Αρχικη σελιδα virtuino)

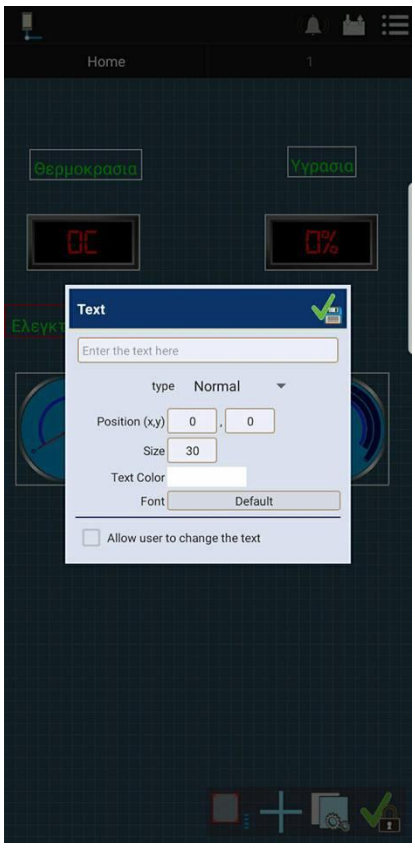
Το πρώτο πραγμα που πρέπει να κάνουμε είναι να προσθέσουμε το κανάλι που έχουμε δημιουργήσει στον thingspeak server
Πιεζουμε αρχικα το εικονίδιο των συνδέσεων→κατω δεξια το +→ επιλέγω (thingspeak channel) ,θα εμφανιστει μια φορμα η οποία θα σας ζητά το "channel id" το οποίο το βρίσκουμε στο thingspeak επισης αλλάζουμε και το "refresh time" (15 sec)





Ξεκινατε και προσθέτετε κάποια <components» όπως τα εχει στην εφαρμογή για τη θερμοκρασία επιλέγουμε το "value display"

Στην επόμενη φορμα που μας εμφανιζει στην επιλογη server βάζουμε το "thinkspeak" το οποιο δημιουργήσαμε προηγουμένως.Τη μεγιστη τιμη της θερμοκρασίας την ορίζουμε στους 50 βαθμούς (end value).
"decimal places"→1
"Read value"→16
Τσεκαρουμε το "display max…"
(Το virtuino σας δίνει τη δυνατότητα να σας ενημερώσει με "sms" αν περασει ένα όριο της θερμοκρασίας που καθοριζουμε εμεις,πατωντας στη καρτέλα sms)



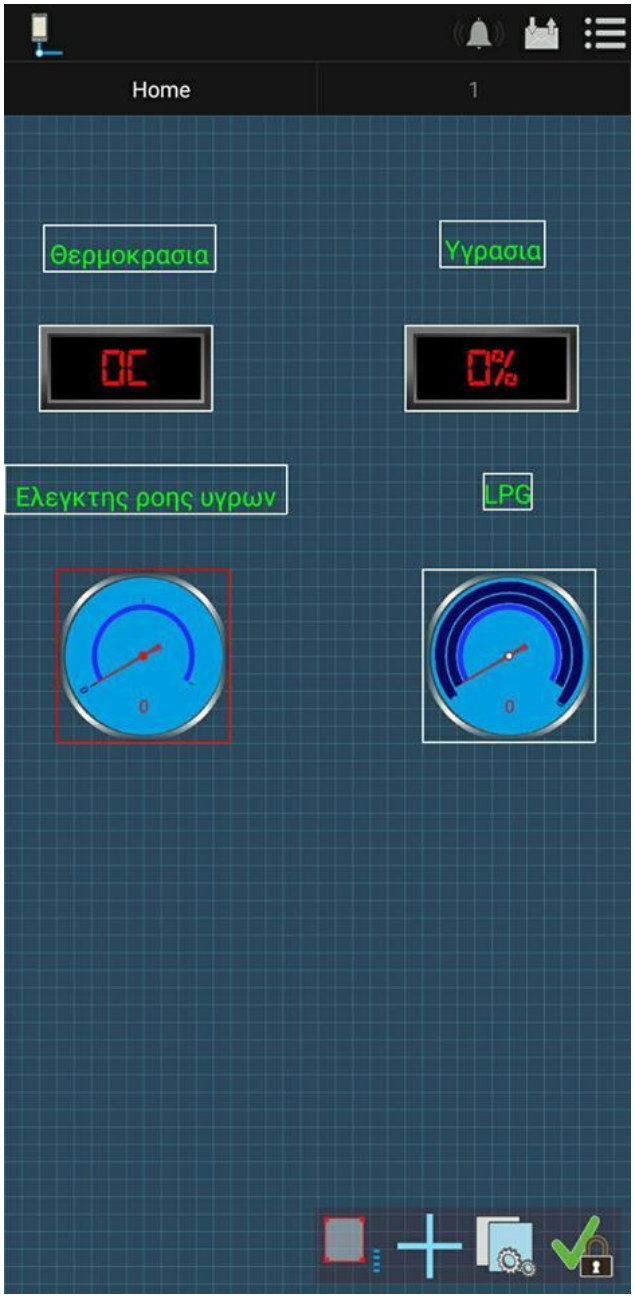
Το σωζετε και προσθέτετε ένα κείμενο διπλα από το μετρητή
Πατατε στο +→text→type(bold)→size(30).
Για την υγρασία θα προσθεσουμε έναν αναλογικό μετρητή

Παταμε +→analog instrument→server(thinkspeak)→analog input(βαζουμε το field οπου εχουμε την υγρασία πχ.field 2)→end value(100)→symbol(%)→backround(μαυρο,μπλε,πρασινο κτλ)→display max..(check)

Επιλέγετε την επόμενη καρτέλα scale.value between scale small line (1)→…middle line (5)→big line(10)→text(20)

Για το κείμενο Πατατε στο +→text→type(bold)→size(40).

Αφου προσθεσαμε ολες τα στοιχεια που θα μας δινουν τις ενδειξεις που θελουμε η τελικη μορφη της εφαρμογης μας θα είναι η εξης:



6.3 Εγκατάσταση προγράμματος ARDUINO IDE

Το πρόγραμμα ARDUINO IDE είναι διαθέσιμο για λειτουργικά WINDOWS,MAC,LINUX 32 η 64bits.Εμείς χρησιμοποιούμε windows λειτουργικό . Θα κατεβάσουμε τη τρέχουσα έκδοση του προγράμματος arduino από το επίσημο site στο ακόλουθο URL

https://www.arduino.cc/en/Main/Software

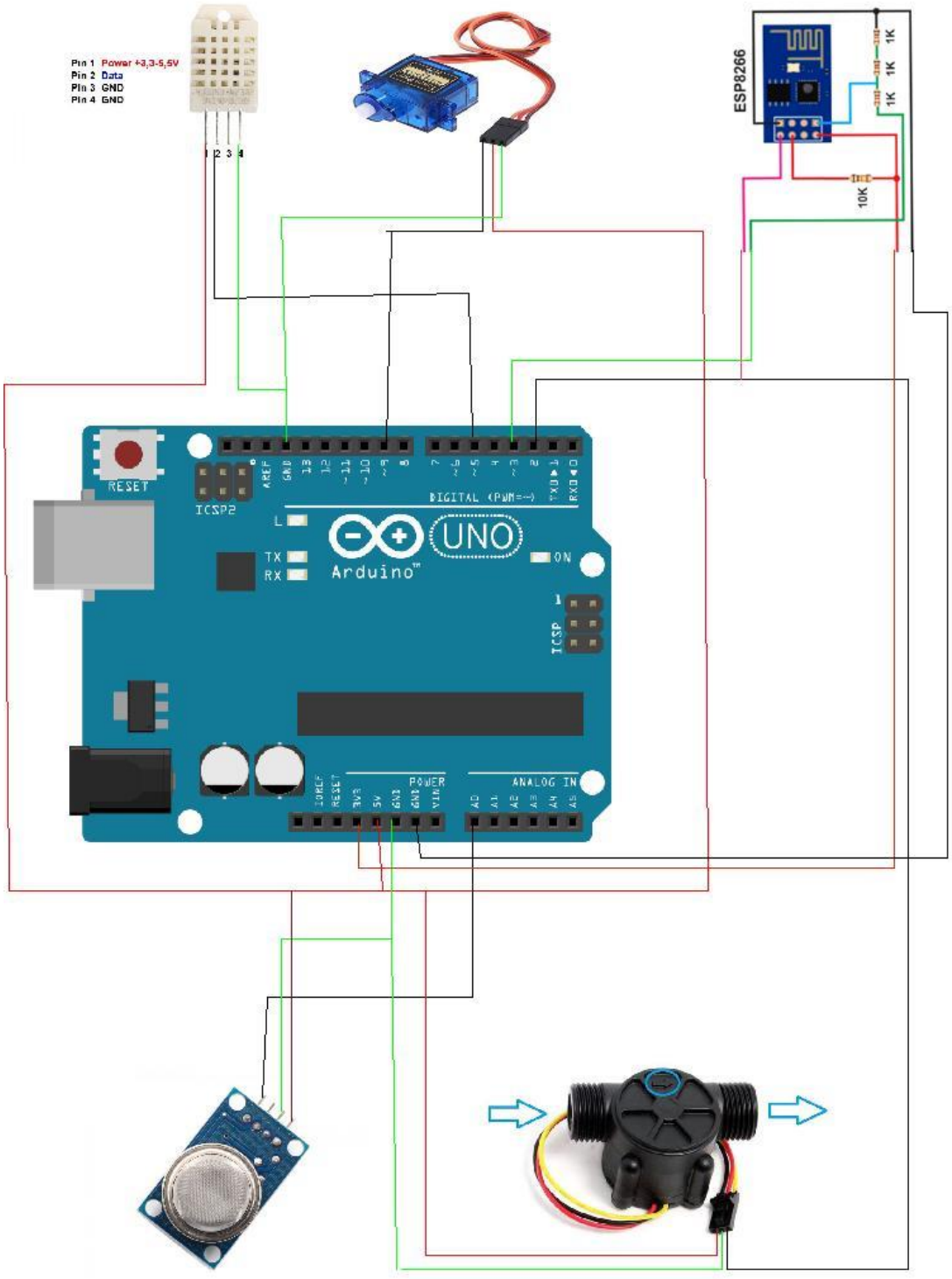
και το συμπιεσμένο αρχείο "arduino 1.8.3"

που φαίνεται στην ακόλουθη εικόνα

Εικόνα 4:download arduino ide

Εφόσον κατεβάσουμε το συμπιεσμένο αρχείο, επειτα πρέπει να το κάνουμε αποσυμπίεση (extract).Μόλις τελειώσει το extract,δε χρειάζεται κάποια άλλη ιδιαίτερη εγκατάσταση και είμαστε έτοιμοι να χρησιμοποιήσουμε το πρόγραμμα μας.Σε παλαιότερες εκδόσεις χρειαζόταν και άλλα βήματα για να κάνεις την εγκατάσταση.Μέσα στο φάκελο υπάρχουν και άλλοι φάκελοι όπως libraries,drivers,examples οπου δε πρέπει να τους πειράξουμε για τη σωστή λειτουργία του προγράμματος.

6.4 Συνδεσμολογία Arduino με αισθητήρια και Esp8266 module



Αφού λοιπόν πραγματοποιήσατε τις συνδέσεις στον arduino,θα πρέπει να φορτώσετε τον κώδικα .

Αρχικα χρειαζόμαστε την εντολή sodtware serial γιατί η επικοινωνία μεταξύ arduino uno και πλακέτας esp8266 γίνεται με μια software σειριακή θυρα στα pin 10,11

```
#include <Servo.h>
                                        //DHT + THINGSPEAK

#include <SoftwareSerial.h>
SoftwareSerial espSerial =  SoftwareSerial(10,11);
```

0

Θα χρειαστεί να κάνουμε μερικές ρυθμίσεις πρέπει να αλλάξουμε το apikey και στη θεση του να βάλουμε το "write api key" που μας δινει το thingspeak.Επίσης αλλάζουμε το ονομα του ασύρματου δικτύου και τον κωδικό.

```
#include <DHT.h>

#define DHTPIN 5
sensor to digital pin 5
#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);

String apiKey = "5G2D0OA5HR80JFT7";     Εδώ γραφουμε το WRITE API key που
μας δινεται από τον σερβερ thingspeak.


String ssid="Monkey D Luffy";    // Wifi network SSID
String password ="Captain1990";  // Wifi  κωδικός


boolean DEBUG=true;
```

1

//SMOKE SENSOR

#define MQ_PIN (0) //Προσδιοριζω ποιο αναλογικο καναλι εισοδου θα χρησιμοποιησω

#define RL_VALUE (5) //Προσδιοριζω την αντισταση που τραβαω όταν ανεβαζω το προγραμμα σε kilo ohms

#define RO_CLEAN_AIR_FACTOR (9.83) //RO_CLEAR_AIR_FACTOR=(Sensor resistance in clean air)/RO,

//Προκυπτει από το datasheet

#define CALIBRATION_SAMPLE_TIMES (50) //Προσδιοριζω ποσα δειγματα θα χρειαστω όταν γινεται το calibration

#define CALIBRATION_SAMPLE_INTERVAL (500) //Προσδιοριζω το χρονο ανακυκλωσης σε millisecond μεταξυ του κάθε δειγματος.

//cablibration phase

#define READ_SAMPLE_INTERVAL (50) //Προσδιοριζω ποσα δειγματα θα παρω σε κανονικη λειτουργια

#define READ_SAMPLE_TIMES (5) //Προσδιοριζω το συνολικο χρονο που θα χρειαστω μεταξυ των δειγματων σε milisecond

#define GAS_LPG (0)

#define GAS_CO (1)

#define GAS_SMOKE (2)

float LPGCurve[3] = {2.3,0.21,-0.47}; //Δυο σημεια παιρνονται από την καμπυλη με αυτά τα δυο σημεια δημιουργειται μια ευθεια που είναι"κατά προσεγγιση ισοδυναμη" με την αρχικη καμπυλη.

```
                                        //data format:{ x, y, slope}; point1: (lg200, 0.21), point2: (lg10000, -0.59)
float          COCurve[3]  =  {2.3,0.72,-0.34};
float          SmokeCurve[3] ={2.3,0.53,-0.44};
float          Ro          =  10;                 //To R0 προσδιοριζεται σε 10 kilo ohms
```

//Αισθητηρας Υγρων

```
byte statusLed    = 13;

byte sensorInterrupt = 0;  // 0 = digital pin 2
byte sensorPin       = 2;

// Ο αισθητηρας υγρων παραγει 4.5 παλμους ανα δευτερολεπτο ανα λιτρο ανα λεπτο ροης

float calibrationFactor = 4.5;

volatile byte pulseCount;

float flowRate;
unsigned int flowMilliLitres;
unsigned long totalMilliLitres;

unsigned long oldTime;
```

// showResponce

```
void showResponse(int waitTime){
    long t=millis();
    char c;
    while (t+waitTime>millis()){
      if (espSerial.available()){                    //DHT22
        c=espSerial.read();
        if (DEBUG)
          Serial.print(c);
      }
    }


}
boolean thingSpeakWrite(float value1, float value2, float value3, int value4, long value5, int value6, int value7, int value8){
  String cmd = "AT+CIPSTART=\"TCP\",\"";                  // TCP connection
  cmd += "184.106.153.149";                              // api.thingspeak.com
  cmd += "\",80";
  espSerial.println(cmd);
  if (DEBUG) Serial.println(cmd);
  if(espSerial.find("Error")){                           //DHT22
    if (DEBUG)
      Serial.println("AT+CIPSTART error");
    return false;
  }
```

```
String getStr = "GET /update?api_key=";   // prepare GET string
getStr += apiKey;

getStr +="&field1=";
getStr += String(value1);
getStr +="&field2=";
getStr += String(value2);
getStr +="&field3=";
getStr += String(value3);
getStr +="&field4=";
getStr += String(value4);
getStr +="&field5=";
getStr += String(value5);
getStr +="&field6=";
getStr += String(value6);
getStr +="&field7=";
getStr += String(value7);
getStr +="&field8=";
getStr += String(value8);
getStr += "\r\n\r\n";
```

```
  cmd = "AT+CIPSEND=";
  cmd += String(getStr.length());
  espSerial.println(cmd);
  if (DEBUG)  Serial.println(cmd);

  delay(100);
  espSerial.print(getStr);
  if (DEBUG)  Serial.print(getStr);

  return true;
}
```

// SMOKE

//Υπολογισμος αντιστασης για το MQ

Εισοδος: raw_adc - raw value read from adc, which represents the voltage

Εξοδος: η υπολογισμενη αντισταση αισθητηρα

Σημειωση: Ο αισθητηρας και και ο αντιστατης φορτωσης δημιουργουν ένα διαιρετη τασης. Δεδομενης της τασης σε ολο τον αντιστατη και της αντιστασης του η αντισταση του αισθητηρα μπορει να παραχθει.

```
//
float MQResistanceCalculation(int raw_adc)
{
  return ( ((float)RL_VALUE*(1023-raw_adc)/raw_adc));
}
```

// MQCalibration

Εισοδος: mq_pin – αναλογικο καναλι

Εξοδος: Ro of the sensor

Σημειωση:Αυτή η λειτουργια υποθετει ότι ο αισθητηρας βρισκεται σε καθαρο αερα.Χρησιμοποιει το MQResistanceCalculation για να υπολογισει την αντισταση του αισθητηρα σε καθαρο αερα και μετα διαιρει με το RO_CLEAN_AIR_FACTOR. Το RO_CLEAN_AIR_FACTOR είναι περιπου 10 το οποιο διαφερει ελαχιστα μεταξυ διαφορων αισθητηρων.

```
float MQCalibration(int mq_pin)

{

  int i;

  float val=0;



  for (i=0;i<CALIBARAION_SAMPLE_TIMES;i++) {            //παιρνει πολλαπλα δειγματα

    val += MQResistanceCalculation(analogRead(mq_pin));

    delay(CALIBRATION_SAMPLE_INTERVAL);

  }

  val = val/CALIBARAION_SAMPLE_TIMES;                  //υπολογιζει τη μεση τιμη



  val = val/RO_CLEAN_AIR_FACTOR;                                  //Διαιρει με RO_CLEAN_AIR_FACTOR

  return val;

}
```

//Αναγνωση του MQ

Εισοδος: mq_pin – αναλογικο καναλι

Εξοδος: Rs του αισθητηρα

Σημειωση: Αυτή η λειτουργια χρησιμοποιει το MQResistanceCalculation για να υπολογισει την αντισταση Rs του αισθητηρα.

Η αντισταση Rs αλλαζει καθως ο αισθητηρας βρισκεται αναμεσα από διαφορετικους στοχους.

gas. Ο χρονος δειγματοληψιας και ο συνολικος εσωτερικος χρονος μεταξυ των δειγματων μπορει να τροποποιηθει αλλαζωντας τον ορισμο των macros.

**********/

```
float MQRead(int mq_pin)
{
  int i;
  float rs=0;

  for (i=0;i<READ_SAMPLE_TIMES;i++) {
    rs += MQResistanceCalculation(analogRead(mq_pin));
    delay(READ_SAMPLE_INTERVAL);
  }

  rs = rs/READ_SAMPLE_TIMES;

  return rs;
}
```

//MQGetGasPercentage

Εισοδος: rs_ro_ratio - Rs διαιρειται απο την Ro

gas_id - τυπος αεριου

Εξοδοςt: ppm του αεριου που μετραμε

Σημειωση: Αυτή η λειτουργια περναει διαφορες καμπυλες στη λειτουργια MQGetPercentage η οποια υπολογιζει τα ppm (parts per million) του αεριου που μετραμε.

```
//
int MQGetGasPercentage(float rs_ro_ratio, int gas_id)
{
  if ( gas_id == GAS_LPG ) {
     return MQGetPercentage(rs_ro_ratio,LPGCurve);
  } else if ( gas_id == GAS_CO ) {
     return MQGetPercentage(rs_ro_ratio,COCurve);
  } else if ( gas_id == GAS_SMOKE ) {
     return MQGetPercentage(rs_ro_ratio,SmokeCurve);
  }

  return 0;
}
```

// MQGetPercentage

Εισοδος: rs_ro_ποσοστο - Rs διαιρειται από την Ro

pcurve - Δεικτης στην καμπυλη του αεριου που παιρνουμε σαν εισοδος

Εξοδος : ppm of the target gas

Σημειωσεις: Χρησιμοποιωντας την κλιση και ένα σημειο στην ευθεια, το σημειο χ(logarithmic value of ppm) της ευθειας μπορει να προκυψει αν το ψ (rs_ro_ratio) μας εχει δωθει. Καθως είναι μια λογαριθμικη συνισταμενη η δυναμη του 10 χρησιμοποιειται για να μετατρεψει το αποτελεσμα σε μη λογαριθμικη τιμη.

```
//

int  MQGetPercentage(float rs_ro_ratio, float *pcurve)

{

  return (pow(10,( ((log(rs_ro_ratio)-pcurve[1])/pcurve[2]) + pcurve[0])));

}
```

// ΥΓΡΟ

```
/*
Ρουτινα διακοπης
 */
void pulseCounter()
{
  // Ανεβαζουμε τον μετρητη παλμων
  pulseCount++;
}

//servo
Servo servo1;
int pos1 = 0;

boolean servoFlag;
boolean servoPrev;
```

// Setup

```
void setup() {

  //LEDS
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);

    servoFlag = true;
    servoPrev = NULL;

    //servo
    servo1.attach(9);

  DEBUG=true;           // ενεργοποιουμε το debug serial
  Serial.begin(9600);
```

//ΚΑΠΝΟΣ

```
  Serial.print("Calibrating...\n");

  Ro = MQCalibration(MQ_PIN);                    //Βαθμονομουμε τον αισθητηρα. Πρεπει να σιγουρεψουμε ότι ο αισθητηρας βρισκεται σε καθαρο αερα όταν γινεται η βαθμονομηση //

  Serial.print("Calibration is done...\n");

  Serial.print("Ro=");

  Serial.print(Ro);

  Serial.print("kohm");

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

//ΥΓΡΑ

```
  // Θετουμε το LED σαν εξοδος.
  pinMode(statusLed, OUTPUT);
  digitalWrite(statusLed, HIGH);  //Εχουμε προσκολλησει ένα  active-low LED
  pinMode(sensorPin, INPUT);
  digitalWrite(sensorPin, HIGH);


  pulseCount        = 0;
  flowRate          = 0.0;
  flowMilliLitres   = 0;
  totalMilliLitres  = 0;
  oldTime           = 0;


  // Ο αισθητηρας του Hall είναι συνδεδεμενος στο pin 2 το οποιο χρησιμοποιει interrupt 0.
  // Είναι κωδικοποιημενο να ενεργοποιειται όταν υπαρξει μεταβαση από HIGH σε LOW (FALLING state change)
  attachInterrupt(sensorInterrupt, pulseCounter, FALLING);
```

```
dht.begin();          //Ενεργοποιηση του DHT22

   espSerial.begin(115200);

   espSerial.println("AT+RST");

  showResponse(1000);
```

Υπαρχει μια πολύ σημαντική εντολή είναι η "esp serialbegin ()" είναι η εντολή που ορίζει την ταχύτητα επικοινωνίας μεταξύ του arduino και του esp8266, το esp8266 εχει μια προκαθορισμένη ταχύτητα 11500 πρεπει να την αλλαξουμε στα 9600

```
espSerial.println("AT+UART_CUR=9600,8,1,0,0");
  espSerial.println("AT+CWMODE=1");

  showResponse(1000);

espSerial.println("AT+CWJAP=\""+ssid+"\",\""+password+"\"");  // ονομα ρουτερ και password

  showResponse(5000);

 if (DEBUG)

    Serial.println("Setup completed");

}
```

//ΕΠΑΝΑΛΗΨΗ

```
void loop() {
```

// ΚΑΠΝΟΣ

```
  double var6 = MQGetGasPercentage(MQRead(MQ_PIN)/Ro,GAS_LPG);
  double var7 = MQGetGasPercentage(MQRead(MQ_PIN)/Ro,GAS_CO);
  double var8 = MQGetGasPercentage(MQRead(MQ_PIN)/Ro,GAS_SMOKE);
```

// ΥΓΡΑ

```
  float var3 = 1;
  int var4 = 1;
  long var5 = 1;


   if((millis() - oldTime) > 1000)    //Επεξεργάζεται μονο τους μετρητες μια κάθε δευτερόλεπτο
  {
    //Διακοπτουμε την ρουτινα διακοπης καθως υπολογιζουμε το ρυθμο ροης και στελνουμε την τιμη στον host
    detachInterrupt(sensorInterrupt);


    // Επειδη αυτος ο βρόγχος μπορει να μην συμπληρωθει σε ακριβως ένα δευτερολεπτο εσωτερικου χρονου υπολογιζουμε τον αριθμο των milliseconds που εχουν περασει από την τελευταια εκτελεση και το χρησιμοποιουμε αυτό για να εξισταθμισουμε το αποτελεσμα. Επισης εφαρμοζουμε ένα παραγοντα βαθμονομησης βασισμενο στον αριθμο παλμων ανα δευτερολεπτο ανα μοναδα μετρησης(λιτρα στην προκειμενη περιπτωση)που ερχονται από τον αισθητηρα.//


    flowRate = ((1000.0 / (millis() - oldTime)) * pulseCount) / calibrationFactor;
```

// Παρατηρουμε την ωρα που εκτελεστηκε αυτή η εντολη. Βλεπουμε ότι επειδη απενεργοποιησαμε τη ρουτινα διακοπης η συναρτηση millis() δε θα αυξηθει αμεσως σε αυτό το σημειο αλλα θα γυρισει στην τιμη που ειχε πριν φυγει η ρουτινα διακοπης.//

oldTime = millis();

//Διαιρω το ρυθμο ροης (λιτρα/λεπτο) με το 60 για να καθορισω ποσα λιτρα περασαν από τον αισθητηρα σε ένα δευτερολεπτο και μετα το πολλαπλασιαζω με το 1000 για να το μετατρεψω σε millilitres.//

flowMilliLitres = (flowRate / 60) * 1000;

//Προσθετω τα millilitres που περασαν σε αυτό το δευτερολεπτο στο συνολο.//

totalMilliLitres += flowMilliLitres;

var3 = flowRate;

var4 = flowMilliLitres;

var5 = totalMilliLitres;

//Επαναφερω το μετρητη παλμων ώστε να μπορω να ξεκινησω να αυξανω ξανα//

pulseCount = 0;

// Ενεργοποιω την ρουτινα διακοπης ξανα τωρα που εχουμε στειλει το αποτελεσμα.//

attachInterrupt(sensorInterrupt, pulseCounter, FALLING);

}

// DHT + THINGSPEAK

```
// Read sensor values
float t = dht.readTemperature();
float h = dht.readHumidity();
if (isnan(t) || isnan(h)) {
  if (DEBUG) Serial.println("Failed to read from DHT");
  t = 1;
  h = 1;
}
if (DEBUG)  Serial.println("Temp="+String(t)+" *C");
if (DEBUG) Serial.println("Humidity="+String(h)+" %");
thingSpeakWrite(t,h,var3,var4,var5,var6,var7,var8);                    //
Γραφω τις τιμες στο thingspeak
        /*Serial.println(var3);
        Serial.println(var4);
        Serial.println(var5);
        Serial.println(var6);
        Serial.println(var7);
        Serial.println(var8);*/
```

//Servo

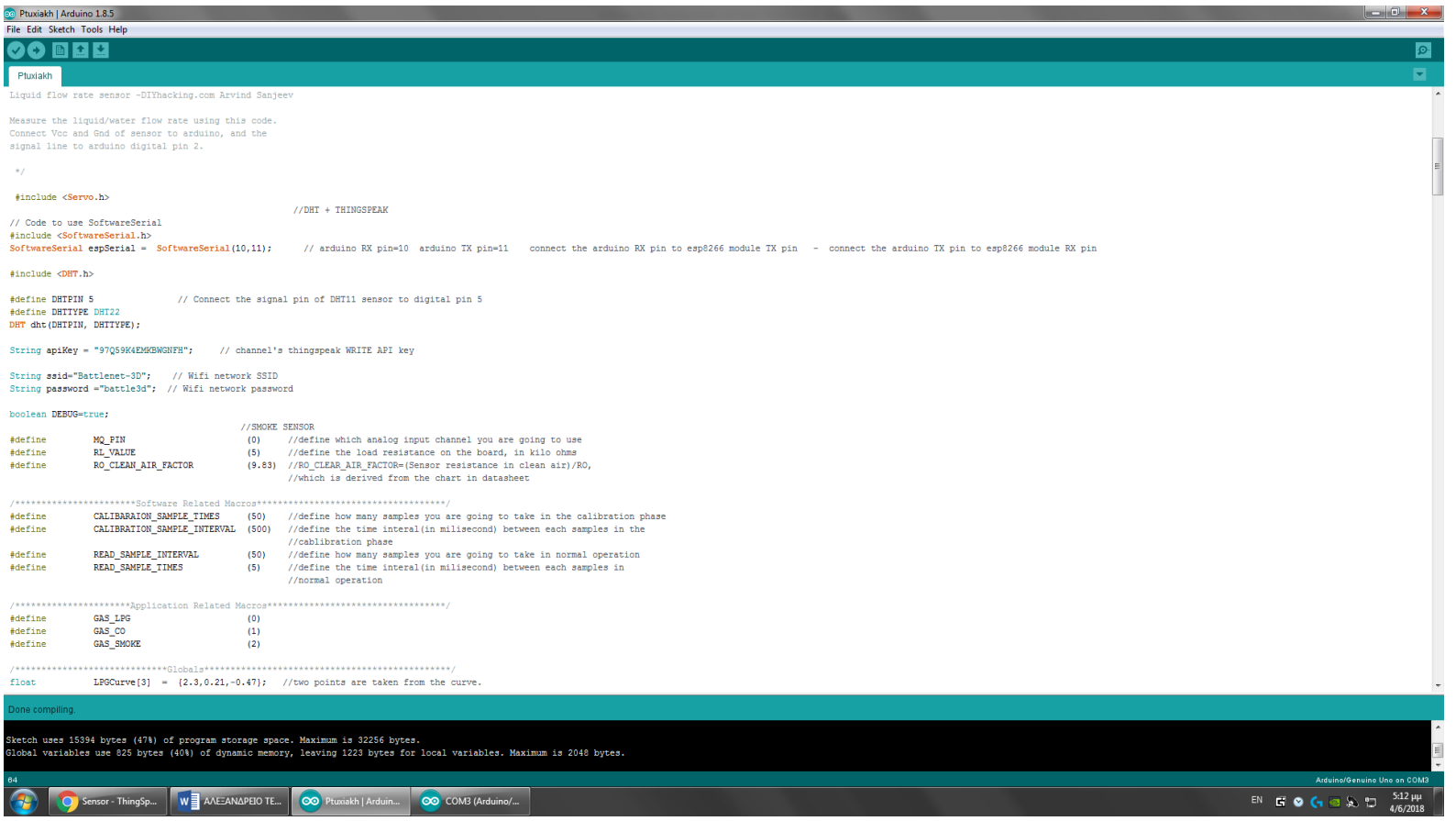
```
servoPrev = servoFlag;
 if(h>90 || t<1 || var3!=0 || var4!=0 || var6!=0 || var7!=0 || var8!=0){
  digitalWrite(12, HIGH);
  digitalWrite(13, LOW);
  Serial.println("FALSE");
  servoFlag = true;
 }
 else{
  digitalWrite(12, LOW);
  digitalWrite(13, HIGH);
  Serial.println("TRUE");
  servoFlag = false;
  }

   if(servoFlag != servoPrev){
    if(servoFlag){
      for(pos1 = 0;pos1<180;pos1 += 1){
       servo1.write(pos1);
       delay(10);
     }
    }
    else{
      for(pos1 = 180; pos1>=1; pos1 -= 1){
      servo1.write(pos1);
       delay(10);
```

```
      }
     }
    }

  // Το thingspeak χρειαζεται 15 sec καθυστερηση μεταξυ των updates,
  delay(20000);
}
```

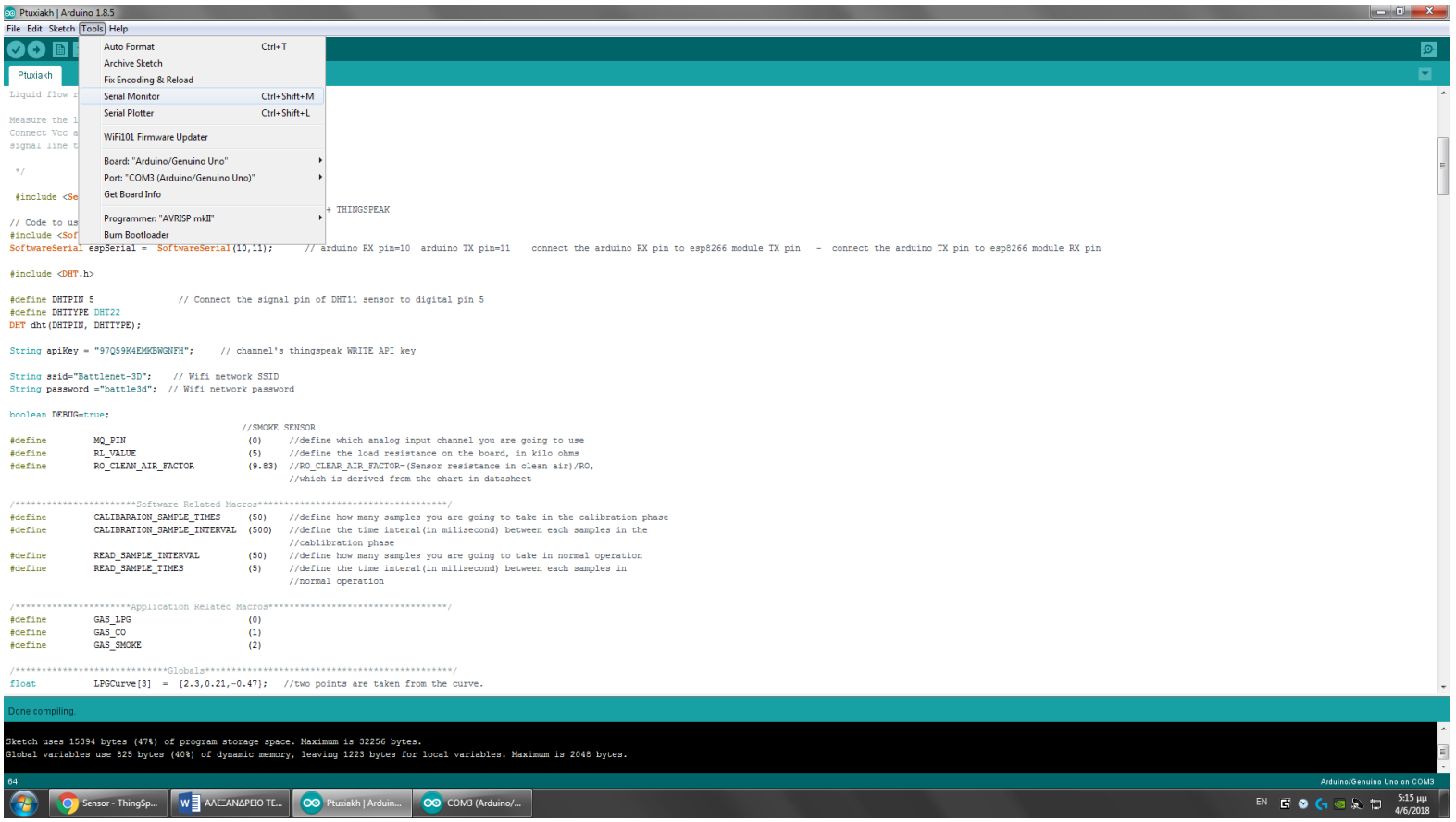
7. Πλήρης λειτουργία κατασκευης

Εφοσον εχουν υλοποιηθει ολες οι απαραιτητες ενεργειες που αναφεραμε παραπανω προχωραμε στη δοκιμη της κατασκευης μας.

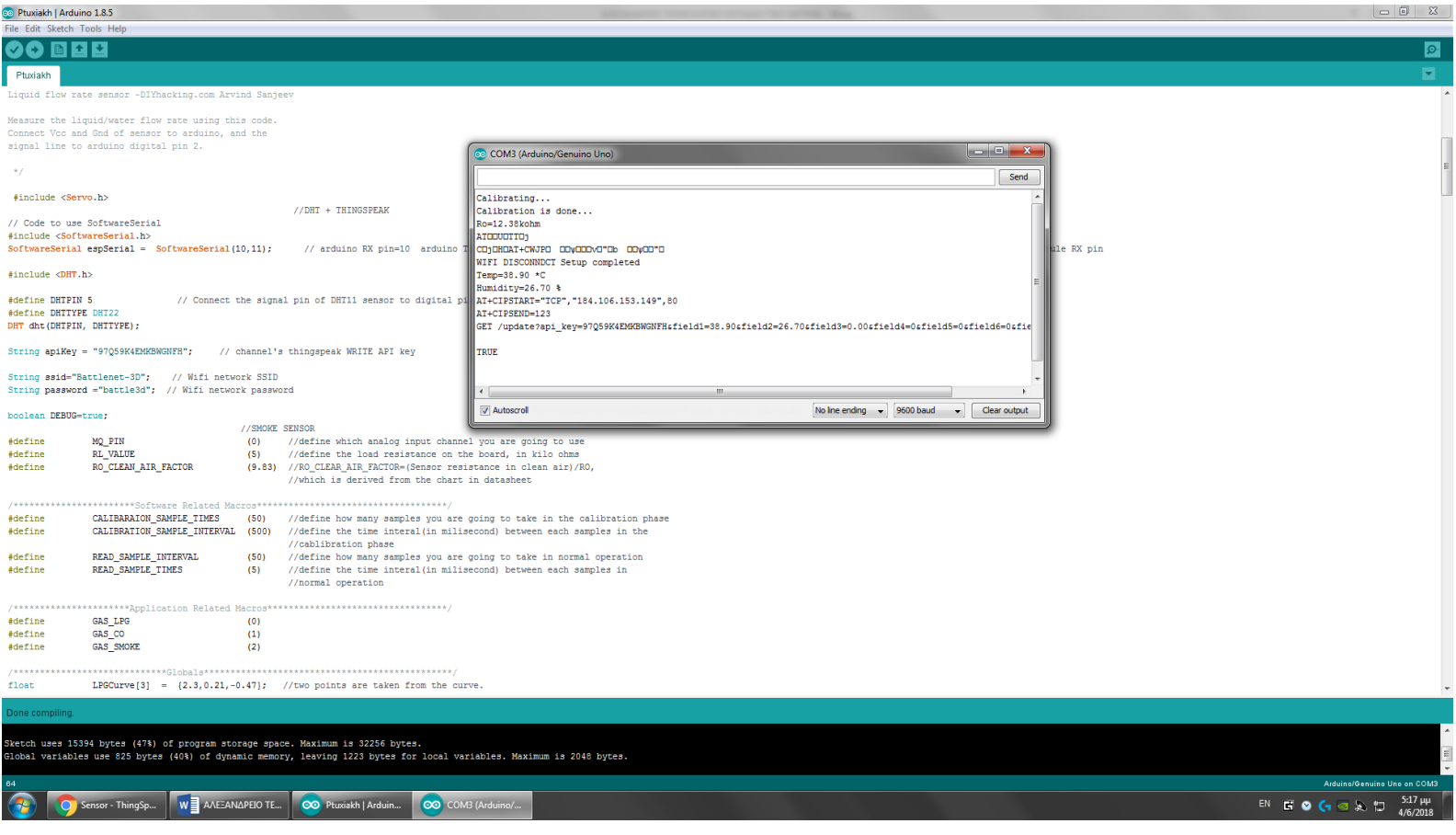
Πρωτο βημα είναι να τσεκαρουμε τον κωδικα. Στο προγραμμα του Arduino πανω αριστερα κατω από τις καρτελες υπαρχει η επιλογη verify. Πατωντας την βλεπουμε αν ο κωδικας μας είναι σωστος.



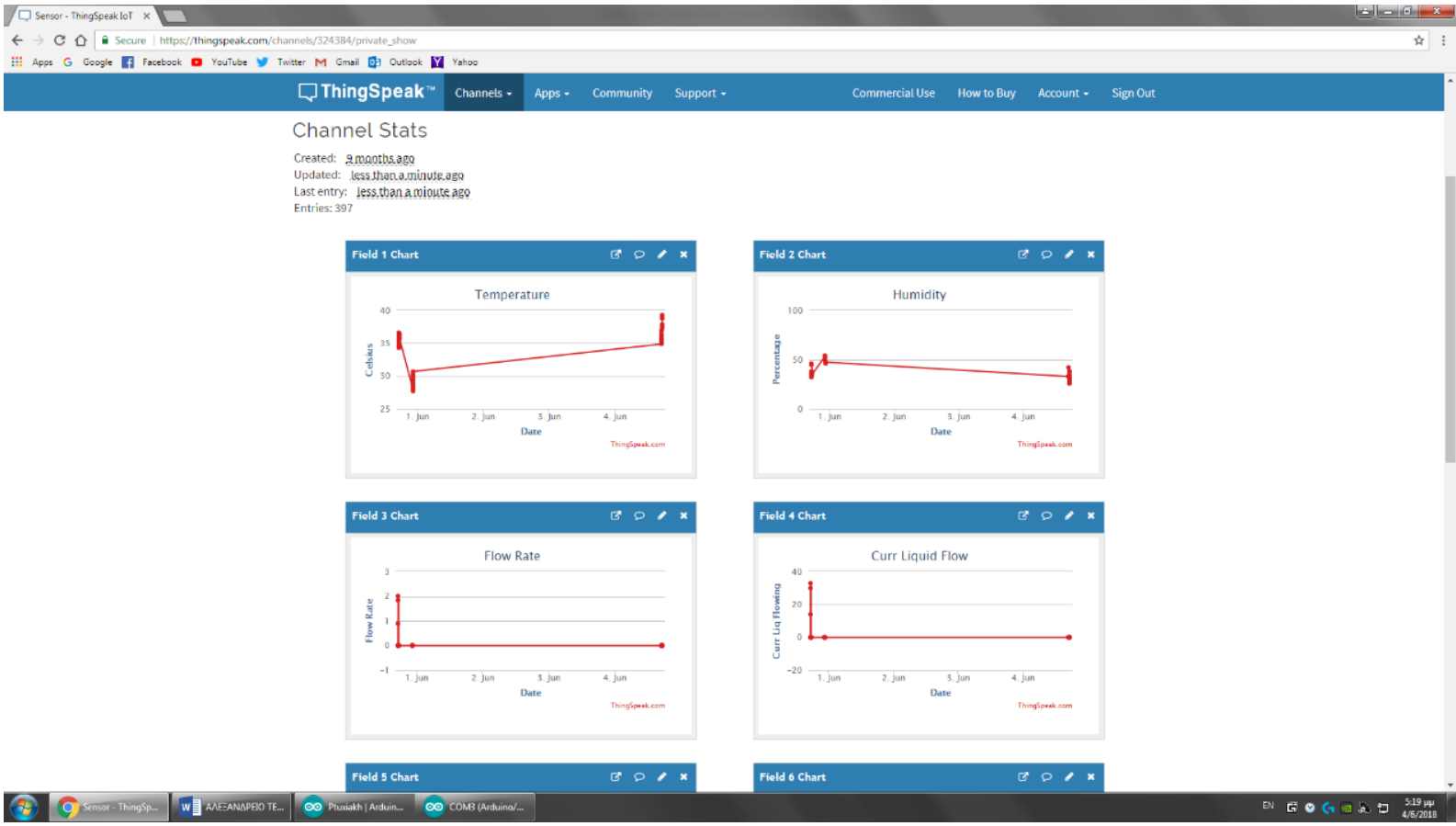
Βλεπουμε ότι ο κωδικας είναι οντως σωστος και προχωραμε στο ανεβασμα του (upload). Το κουμπι βρισκεται ακριβως διπλα από την επιλογη verify. Αφου ανεβασουμε τον κωδικα ανοιγουμε το σειριακο παρακολουθητη (serial monitor). Βρισκεται στην καρτελα εργαλεια.

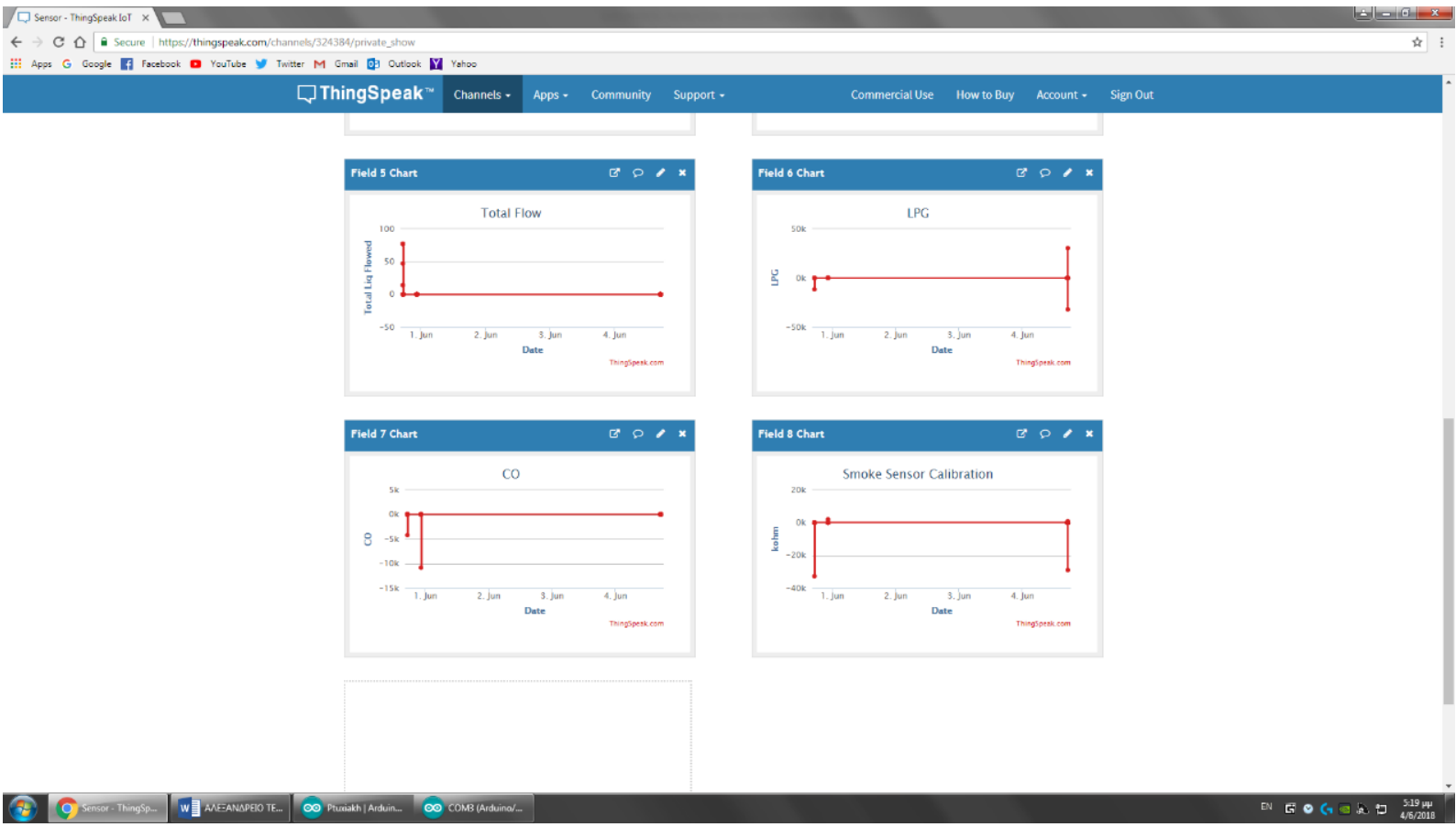


Αφου το προγραμμα μας ανεβει επιτυχως θα δουμε ότι το monitor μας δινει ολες τις τιμες που θελουμε.



Στη συνεχεια ανοιγουμε τη σελιδα του thingspeak και βλεπουμε τις μετρησεις μας διαδικτυακα.





Πραγματοποιώντας διάφορες δοκιμές αναφορικα με τις τιμες που αφορουν τη ροη υδατων από τον αισθητηρα ροης και διαφορες τιμες που αφορουν τον αισθητηρα LPG βλεπουμε ότι η κατασκευη μας λειτουργει κανονικα χωρις κανενα προβλημα και μπορουμε να παιρνουμε αναπασα στιγμη πληροφοριες σχετικα με τη Θερμοκρασια την Υγρασια και τυχον διαρροες υδατων ή αεριων στο χωρο που εχουμε εγκαταστησει την κατασκευη μας.

8. Υλικά κατασκευής

Τα υλικά που θα χρειαστούμε είναι :

1) Arduino uno

2) Esp8266 module

3) αισθητήρα dht22

4) Servo Micro Motor 9G SG90

5) Αισθητήρας αερίου MQ-2

6) Αισθητήριο μέτρησης ροής νερού

7) 2 αντιστάσεις 220ohm for leds

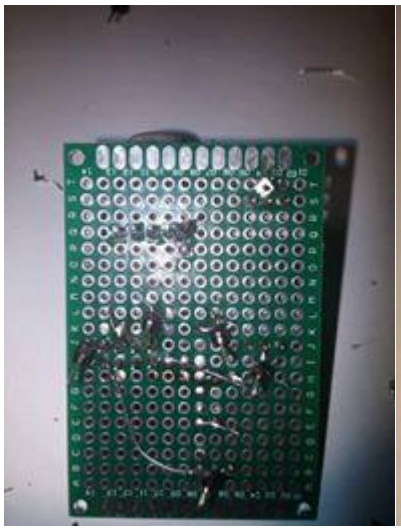
8) 3 αντιστάσεις 1K ohm for Esp8266

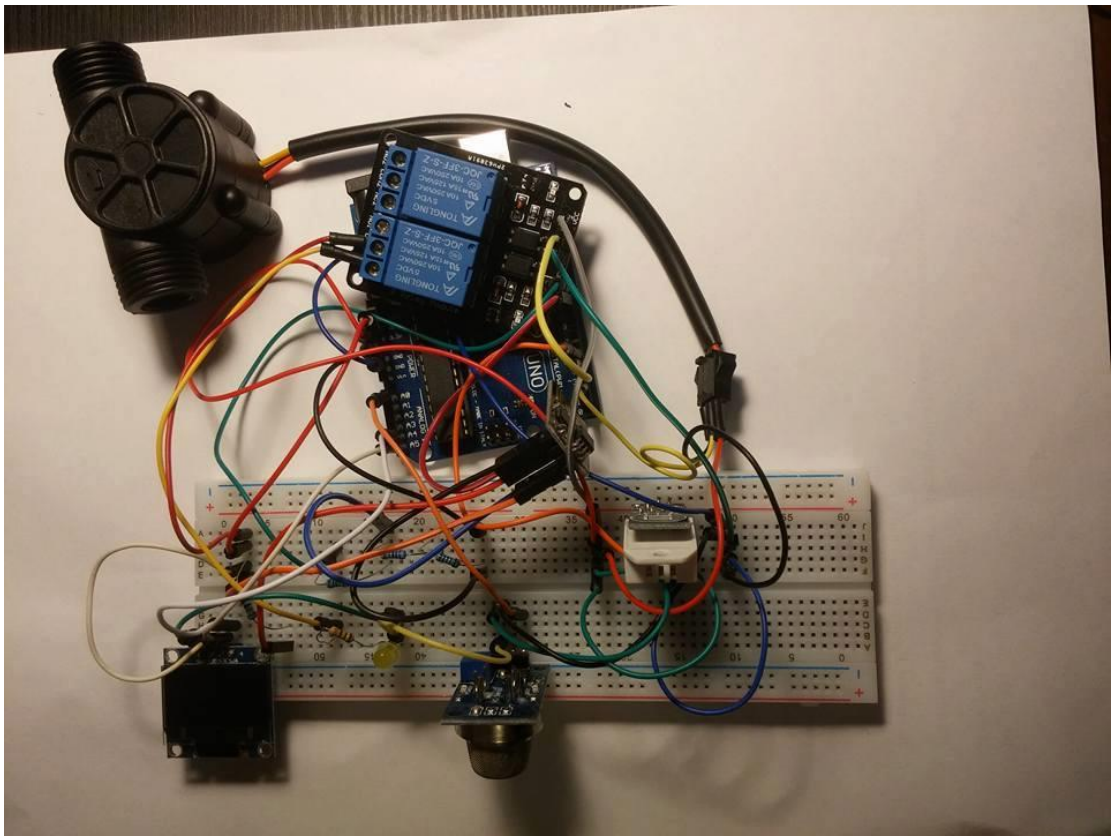
9) 1 αντίσταση 10K ohm for Esp8266

10) 2 led 5V (πράσινο,κόκκινο)

9. Κατασκευαστικό Μέρος






TONGLING
UNO