ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
ΠΜΣ «ΕΠΙΣΤΗΜΗ ΥΠΟΛΟΓΙΣΤΩΝ»

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Μελέτη και ενδεικτική υλοποίηση έξυπνου σπιτιού» (Smart home study and sample implementation)

Ασημακόπουλος Φώτιος

Α.Μ.: 2022201802002

Επιβλέπων:
Καθηγητής Βασιλάκης Κωνσταντίνος

Τρίπολη, Ιανουάριος 2020

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
ΠΜΣ "ΕΠΙΣΤΗΜΗ ΥΠΟΛΟΓΙΣΤΩΝ"

Η παρούσα διπλωματική εργασία παρουσιάστηκε

από τον

Ασημακόπουλο Φώτιο

Α.Μ.: 2022201802002

Την 4η Φεβρουαρίου 2020

«ΜΕΛΕΤΗ ΚΑΙ ΕΝΔΕΙΚΤΙΚΗ ΥΛΟΠΟΙΗΣΗ ΕΞΥΠΝΟΥ ΣΠΙΤΙΟΥ»

ΑΣΗΜΑΚΟΠΟΥΛΟΣ ΦΩΤΙΟΣ

ΠΕΡΙΛΗΨΗ

Σε μία εποχή που η εξέλιξη της τεχνολογίας έχει κάνει μεγάλα βήματα για την επίλυση σημαντικών προβλημάτων, οι οικιακοί αυτοματισμοί δεν αποτελούν εξαίρεση αυτής της ανάπτυξης. Κύριος σκοπός της παρούσας διπλωματικής εργασίας είναι να εισαγάγει τον αναγνώστη σε μεθόδους και τεχνολογίες που έχουν ενσωματωθεί για τη δημιουργία ενός «Έξυπνου Σπιτιού». Επιπρόσθετα, δίνεται έμφαση στις τεχνικές διαχείρισης των κεντρικότερων οικιακών πτυχών όπως θέρμανση – ψύξη, έλεγχος ηλεκτρικών συσκευών, ασφάλεια κατοικίας, με γνώμονα τη διευκόλυνση της διαχείρισης τους από τον χρήστη. Τέλος, στόχος μας είναι να σκιαγραφήσουμε τα κύρια χαρακτηριστικά και οφέλη των Έξυπνων Σπιτιών ως συστατικά στοιχεία ενός καινοτόμου σχεδίου εξέλιξης και διευκόλυνσης της οικιακής ζωής, άλλα και κατασκευής τους με υλικά και τεχνολογία που διαθέτουν υψηλό βαθμό αξιοπιστίας και ταυτόχρονα χαμηλό κόστος.

Λέξεις κλειδιά

Έξυπνο σπίτι, αυτοματισμός, αισθητήρες, έλεγχος μέσω διαδικτύου, μικροεπεξεργαστής, τεχνολογία χαμηλού κόστους.

«SMART HOME STUDY AND SAMPLE IMPLEMENTATION»

ASSIMAKOPOULOS FOTIOS

ABSTRACT

During the past few years, technological developments have contributed towards the provision of solutions to problems in many areas, and the area of home automation has followed along this trend. The main goal of this thesis is to provide for the reader and introduction to methods and technologies that had contribute to the concept of the "Smart Home". Additionally, emphasis is placed on the techniques for managing devices around the most focal concepts of home living, such as heating and air conditioning, electric domestic appliances and home security, aiming to facilitate their management by the user. Finally, this thesis aims to outline the salient characteristics and benefits of the “Smart home”, under the perspective of an innovative plan to advance and facilitate home living. The use of material and technology having a high degree of maturity and dependability as well as low cost are also important parameters that are considered.

Keywords

Smart home, automation, sensors, online control, microprocessor, low cost technology

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ ΟΙΚΟΝΟΜΙΚΗΣ ΑΝΑΛΥΣΗΣ ΚΥΚΛΩΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΙΚΩΝ ΑΝΑΠΑΡΑΣΤΑΣΕΩΝ ΚΥΚΛΩΜΑΤΩΝ ΚΑΙ ΕΙΚΟΝΩΝ

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΧΑΡΙΣΤΙΕΣ

Με την παρούσα διπλωματική εργασία, ολοκληρώνεται η φοίτησή μου στο μεταπτυχιακό πρόγραμμα « Επιστήμη Υπολογιστών », του τμήματος Πληροφορικής και Τηλεπικοινωνιών, του Πανεπιστημίου Πελοποννήσου.

Θα ήθελα να εκφράσω τις ευχαριστίες μου στους καθηγητές όλων των θεματικών ενοτήτων, για τις πολύτιμες γνώσεις που απέκτησα. Ιδιαίτερα θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή της διπλωματικής μου εργασίας **κ. Βασιλάκη Κωνσταντίνο** για την εμπιστοσύνη που μου έδειξε καθώς και για τη συνεχή του καθοδήγηση.

Τέλος θα ήθελα να ευχαριστήσω την οικογένειά μου για τη διαρκή της υποστήριξη και υπομονή που έδειξαν όλα τα μέλη της καθ' όλη τη διάρκεια των σπουδών μου.

1. ΕΙΣΑΓΩΓΗ

1.1 ΒΑΣΙΚΑ ΣΤΟΙΧΕΙΑ ΕΝΟΣ «ΈΞΥΠΝΟΥ ΣΠΙΤΙΟΥ»

Πολλές φορές έχουμε ακούσει τον όρο «Έξυπνο Σπίτι» και θα έχουμε αναρωτηθεί ποιοι παράγοντες καθιστούν ένα σπίτι ως «Έξυπνο».

Η απάντηση στην παραπάνω αναζήτηση δεν είναι άλλη από αυτή ότι ως «Έξυπνο» μπορεί να χαρακτηριστεί ένα σπίτι που παρέχει τις προϋποθέσεις για έναν πιο απλοποιημένο και αναβαθμισμένο τρόπο ζωής του κατόχου σε αυτό. Επομένως αναζητούμε ένα σύνολο μηχανισμών και εφαρμογών που θα έχουν ως αποτέλεσμα την αυτοματοποίηση και τον έλεγχο του.

Όμως ο βαθμός αυτοματοποίησης και ελέγχου ποικίλει, αφού είναι συνάρτηση παραμέτρων όπως είναι το κόστος εφαρμογής καθώς και οι προσωπικές επιθυμίες του κατόχου. Ανεξάρτητα όμως από αυτά ένα «Έξυπνο Σπίτι» αποσκοπεί στο να εξασφαλίζει στους κατοίκους του:

- **Ασφάλεια**, που θα προέρχεται όπως από τη διακοπή ηλεκτροδότησης σε συσκευές όπως ηλεκτρική κουζίνα, θερμοσίφωνο και άλλες ηλεκτρικές συσκευές, αλλά και ενεργοποίηση συστήματος συναγερμού σε περίπτωση διάρρηξης κ.ά.,
- **Άνεση**, όπως να μπορεί ο ιδιοκτήτης και ενεργοποιεί και να απενεργοποιεί τον φωτισμό σε κάποιο χώρο ή και τον κλιματισμό ανάλογα με τις συνθήκες που θέλει να επικρατούν σε κάποιο χώρο του σπιτιού,
- **Εξοικονόμηση ενέργειας**, έχοντας τον έλεγχο ηλεκτρικών συσκευών, της παροχής νερού κ.ά.

Όμως, όπως αναφέρθηκε και ανωτέρω, ένας βασικός παράγοντας δημιουργίας ενός «Έξυπνου Σπιτιού» είναι το και το κόστος κατασκευής. Ως προς αυτό το ζήτημα, η ραγδαία εξέλιξη της τεχνολογίας έχει συνδράμει στη σταδιακή μείωση του κόστους και στο να καταστεί το έξυπνο σπίτι πλέον προσιτό σε ευρεία μερίδα του πληθυσμού.

1.2 ΣΥΝΤΟΜΗ ΠΕΡΙΓΡΑΦΗ ΤΟΥ ΣΚΟΠΟΥ ΚΑΙ ΤΩΝ ΥΠΟΣΥΣΤΗΜΑΤΩΝ

Η παρούσα εργασία έχει ως σκοπό την παρουσίαση και ανάπτυξη ενός συστήματος σε κλίμακα, που η εφαρμογή του στον πραγματικό κόσμο είναι δυνατό να μετατρέψει ένα σπίτι σε «Έξυπνο Σπίτι», με γνώμονα πάντα το χαμηλό κόστος, που εξασφαλίζεται με την κατασκευή ανεξάρτητων ασύρματων κόμβων χαμηλού κόστους (λόγω της χρήσης μικροελεγκτών, αισθητήρων, λοιπών περιφερειακών χαμηλής αγοραστικής αξίας, αλλά και χαμηλού κόστους συντήρησης). Επίσης με την ανάπτυξη ενός απλού σε λειτουργία Πίνακα Ελέγχου, μπορεί ο χρήστης και έχει εύκολη αλληλεπίδραση με το σύνολο των υποσυστημάτων.

Για την ολοκλήρωση του συστήματος αναπτύχθηκαν τα ακόλουθα υποσυστήματα:

- **Σύστημα Ελέγχου Πόρτας Γκαράζ,** όπου ένας μικροελεγκτής ESP32 προγραμματίστηκε ώστε να ελέγχει ένα διπλό ρελέ και ανάλογα με τις ενδείξεις δύο μαγνητικών επαφών, να μπορεί να θέσει σε λειτουργία ένα μοτέρ και να μπορεί ο χρήστης μέσω διαδικτύου να μπορεί να γνωρίζει την κατάσταση της πόρτας του γκαράζ του και να ενεργεί ανάλογα με την επιθυμία του ως προς το άνοιγμα ή το κλείσιμο της.
- **Σύστημα Ελέγχου Φωτισμού Δωματίου,** όπου με χρήση μικροελεγκτή ESP32 ο οποίος ελέγχει ένα ρελέ και ένα photoresistor και βρίσκεται συνδεδεμένος στο ίδιο κύκλωμα με διακόπτη που δίνει σήμα σε έναν δεύτερο ρελέ, μπορεί ο χρήστης είτε μέσω διαδικτύου είτε με τη φυσική του αλληλεπίδραση με το σύστημα να γνωρίζει και να ελέγχει τον φωτισμό ενός δωματίου.
- **Σύστημα Ελέγχου Χώρου,** όπου με χρήση ενός μικροελεγκτή ESP32 CAM, ο χρήστης μπορεί να έχει εικόνα σε πραγματικό χρόνο της περιοχής μπροστά από τον μικροελεγκτή, αλλά και με την εγκατάσταση ενός δεύτερου μικροελεγκτή του ιδίου τύπου και ο οποίος είναι συνδεδεμένος με ανιχνευτή κίνησης HC-SR501, μπορεί να έχει φωτογραφίες της περιοχής σε περίπτωση ενεργοποίησης του παραπάνω ανιχνευτή. Στο σύστημα ελέγχου χώρου μπορεί να ενσωματωθεί και λειτουργία ηχητικής σειρήνας και ενδεχομένως και ενημέρωσης του χρήστη, μετατρέποντάς το έτσι σε σύστημα συναγερμού.
- **Σύστημα Ελέγχου Κλιματισμού,** όπου μικροελεγκτής ESP32 ο οποίος έχει προγραμματιστεί να ελέγχει ένα ρελέ, έναν αισθητήρα τάσης και έναν αισθητήρα DHT11, βρίσκεται συνδεδεμένος στο ίδιο κύκλωμα με διακόπτη που δίνει σήμα σε έναν δεύτερο ρελέ, δίνει τη δυνατότητα στο χρήστη είτε μέσω διαδικτύου είτε με τη φυσική του αλληλεπίδραση με το σύστημα να γνωρίζει την κατάσταση του κλιματισμού ενός δωματίου καθώς και να τον ελέγχει. Το συγκεκριμένο σύστημα μπορεί πολύ εύκολα να εφαρμοστεί

και για τον έλεγχο κατάστασης και άλλων ηλεκτρικών συσκευών όπως π.χ. ηλεκτρική κουζίνα, θερμοσίφωνο κ.ά.

- **Σύστημα Ελέγχου Τιμών Περιβαλλοντικών Μεταβλητών,** όπου μικροελεγκτής ESP32 που βρίσκεται συνδεδεμένος με αισθητήρα DHT22 και BMP180, παρέχει στον χρήστη διαδικτυακά και σε πραγματικό χρόνο, πληροφορίες που αφορούν τη θερμοκρασία και την υγρασία της ατμόσφαιρας, καθώς και την τιμή της ατμοσφαιρικής πίεσης.

1.3 ΔΟΜΗ ΕΡΓΑΣΙΑΣ

Στα κεφάλαια που ακολουθούν γίνεται αναλυτική περιγραφή όλων όσων απαιτούνται για την κατανόηση και τη δημιουργία των παραπάνω υποσυστημάτων. Ειδικότερα:

- Στο κεφάλαιο 2 παρουσιάζονται πληροφορίες σχετικά με:
  - τις πλατφόρμες ανάπτυξης του απαραίτητου λογισμικού (εγκατάσταση και λειτουργικότητα),
  - τον τρόπο εγκατάστασης του απαραίτητου Firmware στους μικροελεγκτές ESP32,
  - τις γλώσσες ανάπτυξης λογισμικού που χρησιμοποιήθηκαν και για την ολοκλήρωση των υποσυστημάτων.
- Στο κεφάλαιο 3 παρουσιάζονται πληροφορίες σχετικά με τους μικροελεγκτές, τους αισθητήρες και τα υπόλοιπα περιφερειακά που απαιτήθηκαν για την ολοκλήρωση των υποσυστημάτων.
- Στο κεφάλαιο 4 γίνεται αναλυτική περιγραφή των υποσυστημάτων τόσο ως προς τη συνδεσμολογία των διαφόρων μερών τους, όσο και ως προς τον τρόπο λειτουργίας τους αλλά και το κόστος κατασκευής τους. Επίσης στο κεφάλαιο αυτό έχουμε και την ανάλυση της λειτουργίας της ιστοσελίδας ελέγχου των υποσυστημάτων.
- Τέλος στα παραρτήματα παρατίθενται πληροφορίες για την εγκατάσταση των προγραμμάτων και εφαρμογών του περιβάλλοντος ανάπτυξης, ενώ δίδονται και τα προγράμματα οδήγησης των διαφόρων υποσυστημάτων.

2. ΤΕΧΝΟΛΟΓΙΚΟ ΥΠΟΒΑΘΡΟ

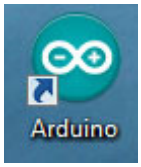
Όταν θέλουμε να προγραμματίσουμε έναν μικροελεγκτή θα πρέπει να επιλέξουμε κατάλληλο περιβάλλον ανάπτυξης, ανάλογα με τον επεξεργαστή αλλά και τη γλώσσα που θα επιλέξουμε για τον προγραμματισμό του. Στην παρούσα ενότητα θα περιγράψουμε τη λειτουργικότητα των κύριων πλατφορμών ανάπτυξης, καθώς και τα βασικά χαρακτηριστικά των γλωσσών προγραμματισμού και σήμανσης που που χρησιμοποιήθηκαν στην εργασία. Τεχνικές λεπτομέρειες για την εγκατάσταση των πλατφορμών δίνονται στο παράρτημα.

2.1 ΠΛΑΤΦΟΡΜΕΣ ΑΝΑΠΤΥΞΗΣ ΛΟΓΙΣΜΙΚΟΥ

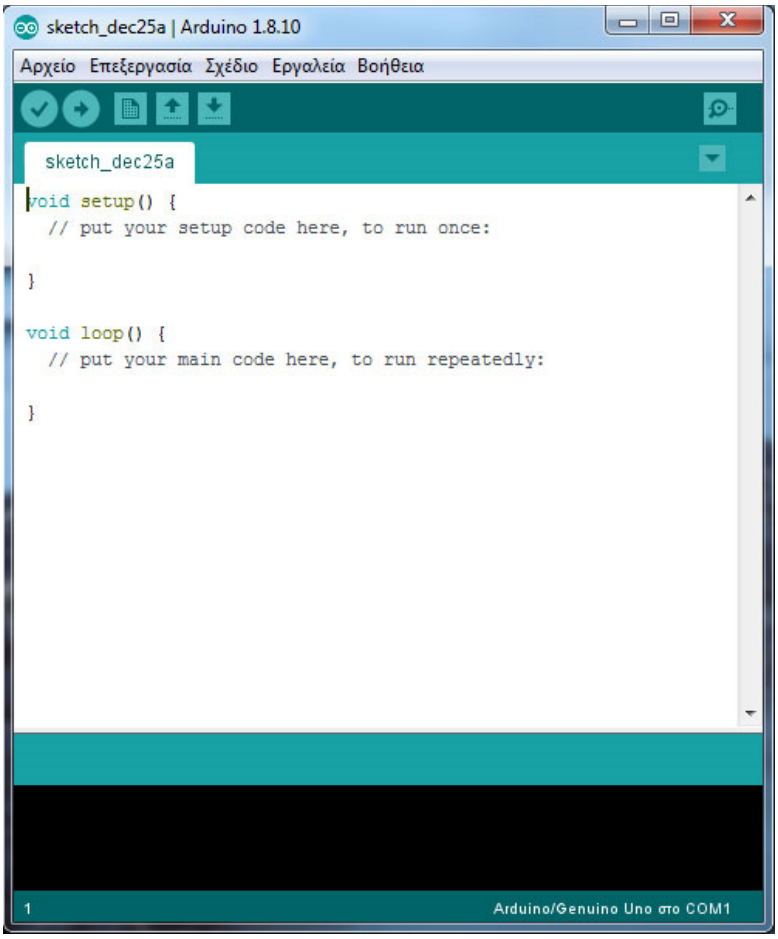
Στην παρούσα εργασία επιλέχθηκαν, η πλατφόρμα Arduino IDE για τον προγραμματισμό των δύο μικροελεγκτών ESP32 CAM και των περιφερειακών τους που χρησιμοποιήθηκαν στη δημιουργία του κυκλώματος ελέγχου του χώρου, ενώ ο προγραμματισμός των μικροελεγκτών ESP32 και των περιφερειακών τους που χρησιμοποιήθηκαν στα υπόλοιπα κυκλώματα έγινε με χρήση της MicroPython χρησιμοποιώντας τις πλατφόρμες Thonny IDE και uPyCraft IDE.

2.1.1 ARDUINO IDE

Μόλις ολοκληρώσουμε την εγκατάσταση του Arduino IDE στον υπολογιστή μας (βλέπε Παραρτήματα - Εγκατάσταση Λογισμικού - Arduino IDE) μπορούμε να προχωρήσουμε στη δημιουργία του πρώτου μας προγράμματος.
Ανοίγουμε το πρόγραμμα εφαρμόζοντας διπλό κλικ επάνω στο εικονίδιο του που έχει δημιουργηθεί από τη διαδικασία εγκατάστασης στην επιφάνεια εργασίας.



Η οθόνη στην οποία οδηγούμαστε είναι η ακόλουθη



Όπως παρατηρούμε μας οδηγεί πως στο εσωτερικό της συνάρτησης setup() τοποθετούμε το τμήμα του κώδικα που θα «τρέξει» μόνο μία φορά κατά την ενεργοποίηση του μικροελεγκτή, ενώ στο εσωτερικό της συνάρτησης loop() τοποθετούμε το τμήμα του κώδικα που θα εκτελείτε επαναλαμβανόμενα.

Κάτω από το μενού επιλογών παρατηρούμε την με τα πέντε εικονίδια αριστερά και ένα δεξιά.

Το πρώτο εικονίδιο  χρησιμοποιείται όταν έχουμε γράψει τον κώδικά μας, προκειμένου να επικυρώσει την ορθότητά του.

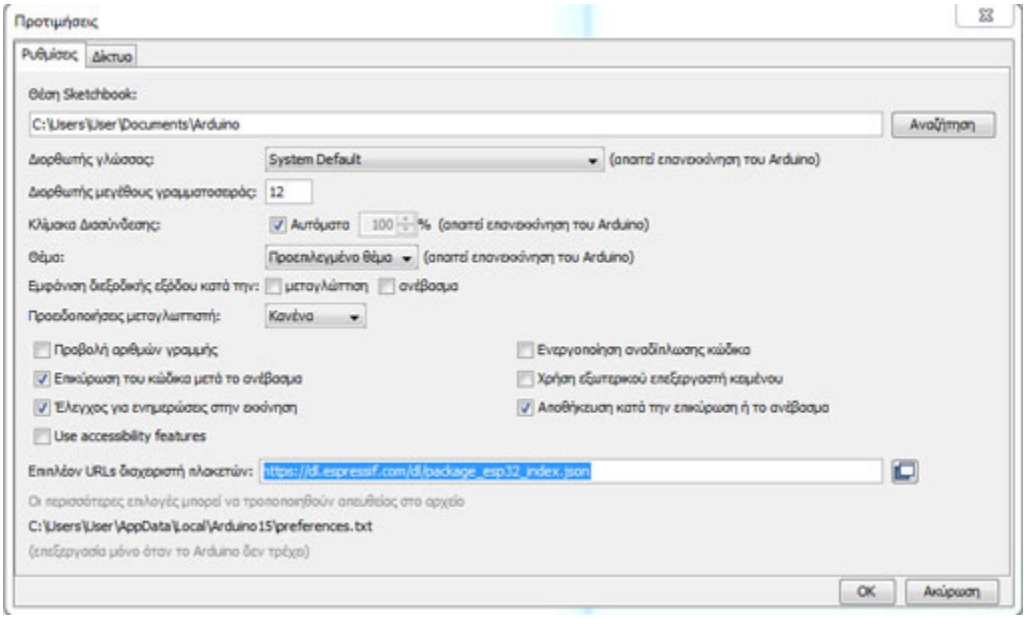
Το δεύτερο εικονίδιο  χρησιμοποιείται όταν θέλουμε να μεταφορτώσουμε τον κώδικα μας στην πλακέτα που στην εργασία είναι το ESP32 CAM. Θα πρέπει ωστόσο να δοθεί προσοχή ώστε να ακολουθηθεί πρώτα η διαδικασία εγκατάστασης που θα περιγράψουμε παρακάτω.

Τα εικονίδια 3-5 ( ,  , ) δίνουν τη δυνατότητα δημιουργίας νέου, ανοίγματος υπάρχοντος και αποθήκευσης προγράμματος, αντίστοιχα. τρίτο εικονίδιο χρησιμοποιείται όταν θέλουμε να δημιουργήσουμε νέο πρόγραμμα.

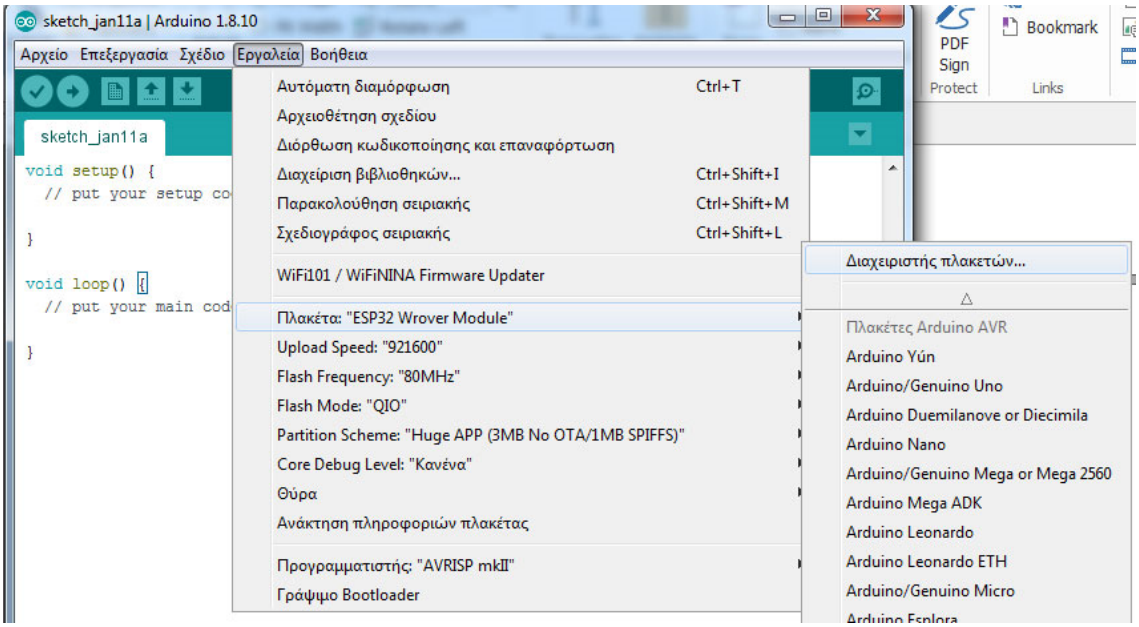
Τέλος το εικονίδιο  που βρίσκεται στα αριστερά της μπάρας χρησιμοποιείται όταν έχουμε ανεβάσει τον κώδικα στην πλακέτα μας και θέλουμε να ανοίξουμε την οθόνη στην οποία εμφανίζονται τα μηνύματα κατά την εκτέλεση του κώδικα μας και που ονομάζεται «σειριακή οθόνη».

Ακολούθως πρέπει να εγκαταστήσουμε την πλακέτα ESP32 CAM. Αυτό το επιτυγχάνουμε ακολουθώντας τα ακόλουθα βήματα:

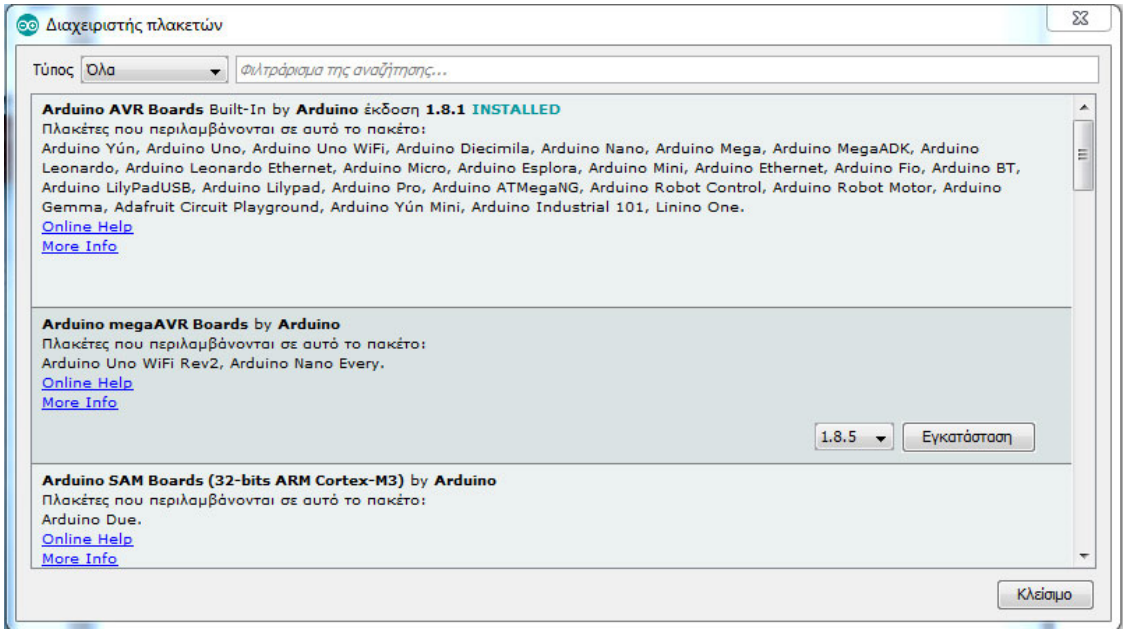
- Στο μενού «Αρχείο» επιλέγουμε «Προτιμήσεις» και στην καρτέλα που ανοίγει εισάγουμε το URL https://dl.espressif.com/dl/package_esp32_index.json στο πεδίο «Επιπλέον URLs διαχειριστή πλακετών» και πατάμε «OK».

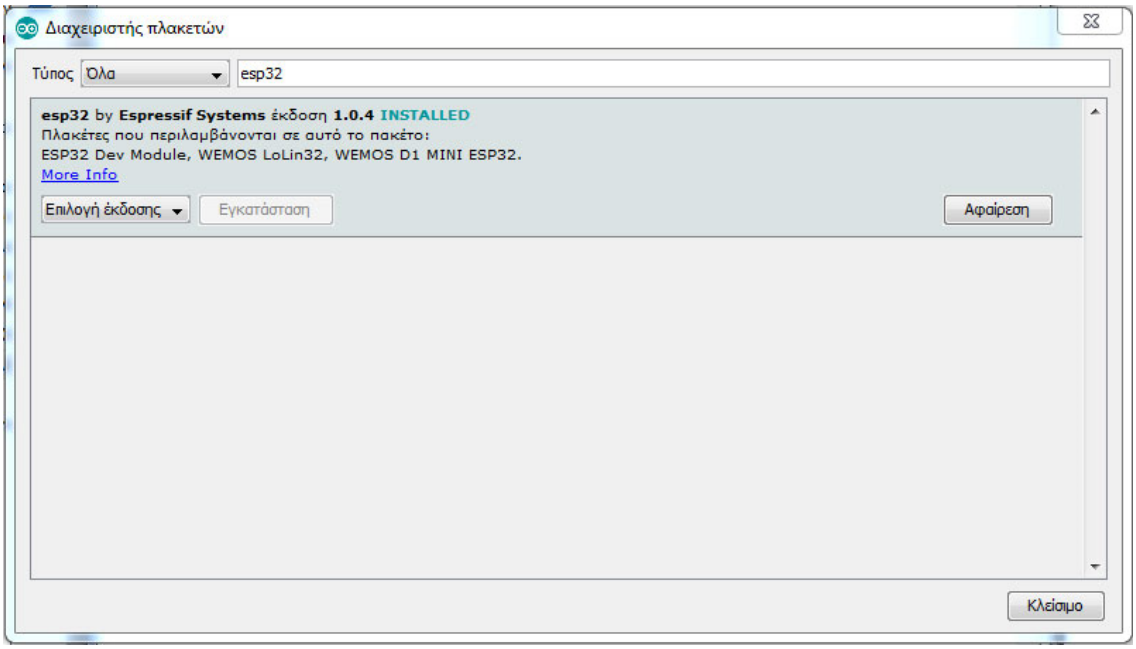


- Κατόπιν στο μενού «Εργαλεία» και στην επιλογή «Πλακέτα:» επιλέγουμε το «Διαχειριστής Πλακετών από το μενού όπου ανοίγει.



- Στην καρτέλα που ανοίγει εντοπίζουμε τον ελεγκτή ESP 32. Ο δόκιμος τρόπος είναι να χρησιμοποιήσουμε το πεδίο «Φιλτράρισμα της αναζήτησης….», εισάγοντας το κείμενο «ESP32». Αφού διαλέξουμε τον ελεγκτή, κατόπιν επιλέγουμε «Εγκατάσταση» και «Κλείσιμο».

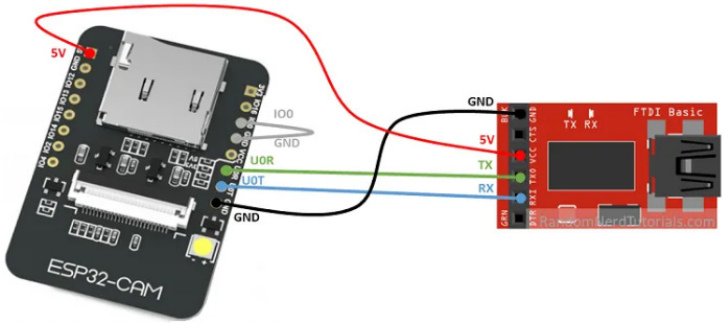




Πλέον το περιβάλλον έχει ετοιμαστεί για την ανάπτυξη κώδικα.

Στο επόμενο βήμα πρέπει να γίνει η προετοιμασία της πλακέτας για να συνδεθεί με τον υπολογιστή. Τα βήματα που πρέπει να ακολουθηθούν είναι τα εξής:

1. Συνδέουμε την πλακέτα ESP32 CAM με την πλακέτα FTDI σύμφωνα με το ακόλουθο διάγραμμα και προσέχουμε να βραχυκυκλώσουμε τα PINs GND και IO0 (γκρι απόχρωση στο σχήμα), διότι σε διαφορετική περίπτωση δεν θα είναι δυνατή η μεταφόρτωση κώδικα στην πλακέτα.



   Στην περίπτωση που η πλακέτα FTDI που χρησιμοποιούμε έχει επιλογή τάσης 3,3V ή 5V φροντίζουμε ο επιλογέας (jumper) να βρίσκεται στην θέση επιλογής των 5V.

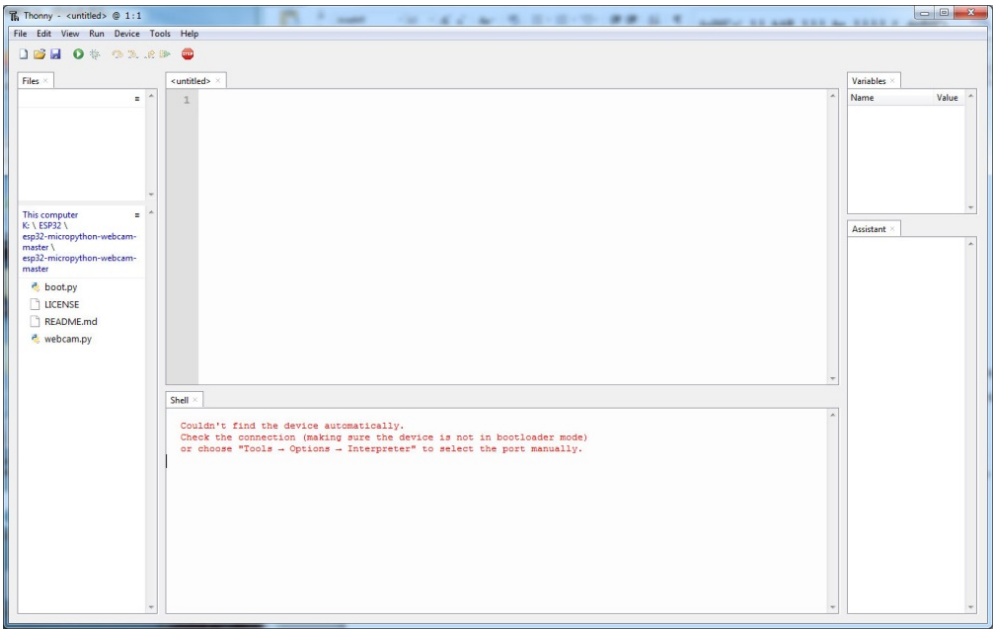
2. Μόλις βρεθούμε στο σημείο όπου θέλουμε να τον ανεβάσουμε στην πλακέτα μας πηγαίνουμε στο μενού «Εργαλεία», στο μενού «Πλακέτα» επιλέγουμε «AI-Thinker ESP32-CAM» και τη «Θύρα» που έχει συνδεθεί η πλακέτα μας.

3. Αμέσως μετά πατούμε το πλήκτρο-εικονίδιο  για να ξεκινήσει το ανέβασμα του κώδικα.
4. Μόλις στο παράθυρο προόδου στο κάτω μέρος του Arduino IDE δούμε την ακόλουθη εικόνα πατάμε το RST (Reset) κουμπί στην πλακέτα ESP32-CAM.
5. Μετά από λίγα λεπτά που τελειώνει το ανέβασμα αποσυνδέουμε το καλώδιο με το οποίο είχαμε βραχυκυκλώσει τα PINs: GND και IO0 και πατάμε το κουμπί RST για να ξεκινήσει η εκτέλεση του κώδικα που ανεβάσαμε στην πλακέτα.

2.2 THONNY IDE

Το Thonny IDE είναι περιβάλλον προγραμματισμού για τις γλώσσες προγραμματισμού Python και MicroPython για τις ανάγκες της εργασίας. Η εγκατάστασή του περιγράφεται στο παράρτημα (ενότητα 7.1.2 - THONNY IDE).

Μόλις ανοίξουμε την εφαρμογή εμφανίζεται το ακόλουθο παράθυρο



Στην αριστερή στήλη βρίσκονται οι λίστες με τα αρχεία στον σκληρό μας δίσκο (στο κάτω μέρος) και όταν συνδέσουμε και τον μικροελεγκτή μας (π.χ. ESP32 στα πλαίσια της εργασίας) τα αρχεία που περιέχει θα φαίνονται πάνω από αυτά του σκληρού μας δίσκου στα οποία εφαρμόζοντας διπλό κλικ από το ποντίκι μας ανοίγουν στο παράθυρο του editor (κεντρική στήλη επάνω παράθυρο).

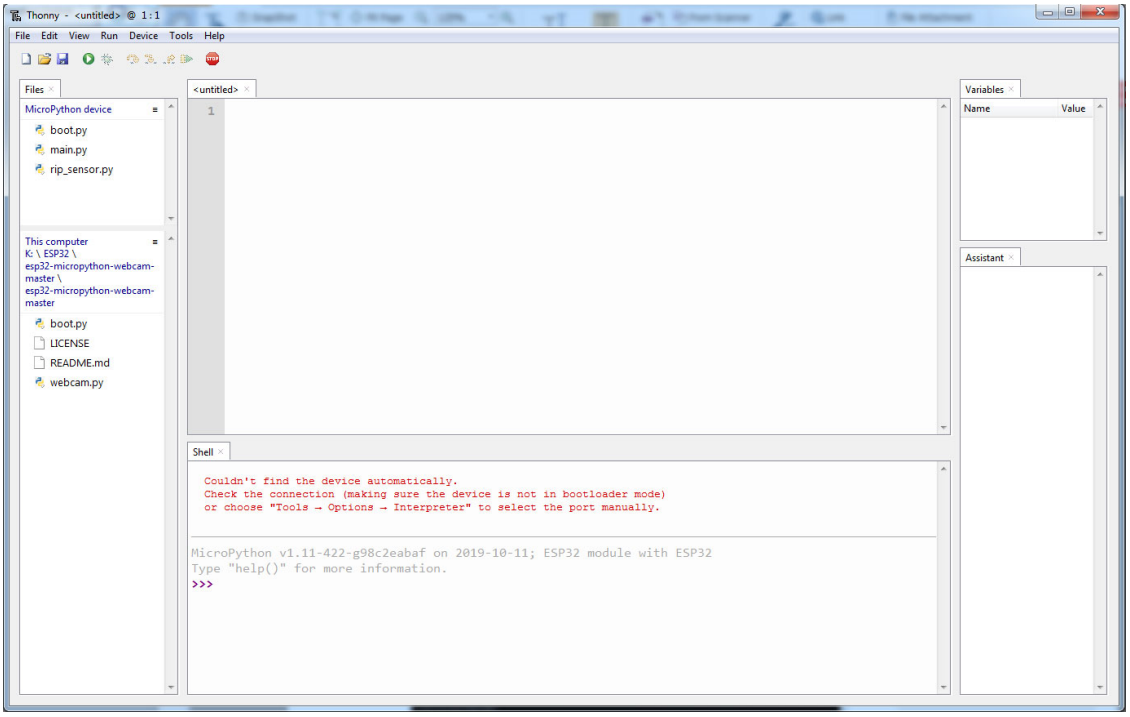
Για να συνδέσουμε τον μικροελεγκτή μας θα πρέπει να εκτελέσουμε πρώτα δύο διαδικασίες.

- Πρώτα θα πρέπει να επιλέξουμε από μενού «**Tools**» το «**Options**» και στην καρτέλα που ανοίγει μεταβαίνουμε στην επιλογή «**Interpreter**» όπου στην επιλογή «**Which interpreter or device should use for running your code?**» επιλέγουμε «**MicroPython (generic)**», ενώ στην επιλογή «**Port**» επιλέγουμε «**Try to detect port automatically**».
- Εγκαθιστούμε το MicroPython firmware (βλέπε ΠΑΡΑΡΤΗΜΑΤΑ – ΕΓΚΑΤΑΣΤΑΣΗ ΛΟΓΙΣΜΙΚΟΥ – ΕΓΚΑΤΑΣΤΑΣΗ FIRMWARE)

Κατόπιν συνδέουμε τον μικροελεγκτή μας μέσω USB και πατάμε το εικονίδιο 

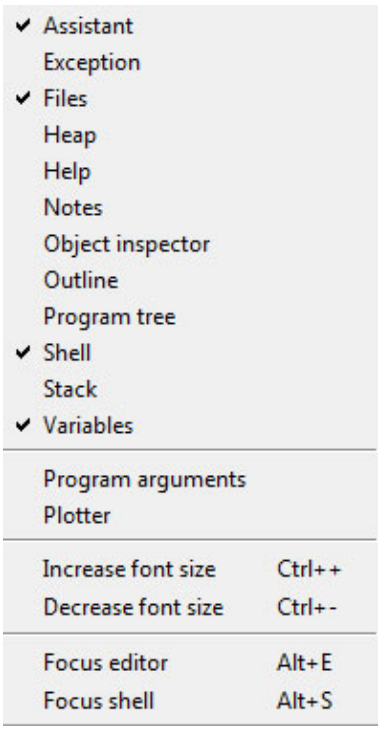
.

Μόλις συνδεθεί ο μικροελεγκτής τότε θα εμφανιστεί η ακόλουθη οθόνη:



Στην κεντρική στήλη έχουμε τα δύο βασικά παράθυρα του προγράμματος, δηλαδή τον editor όπου μπορούμε να γραφούμε τον κώδικά μας και το Shell όπου μπορούμε να βλέπουμε τα αποτελέσματα του κώδικα αλλά και να γράψουμε εντολές όπου θα εκτελούνται η κάθε μία ξεχωριστά βλέποντας αμέσως το αποτέλεσμα τους.

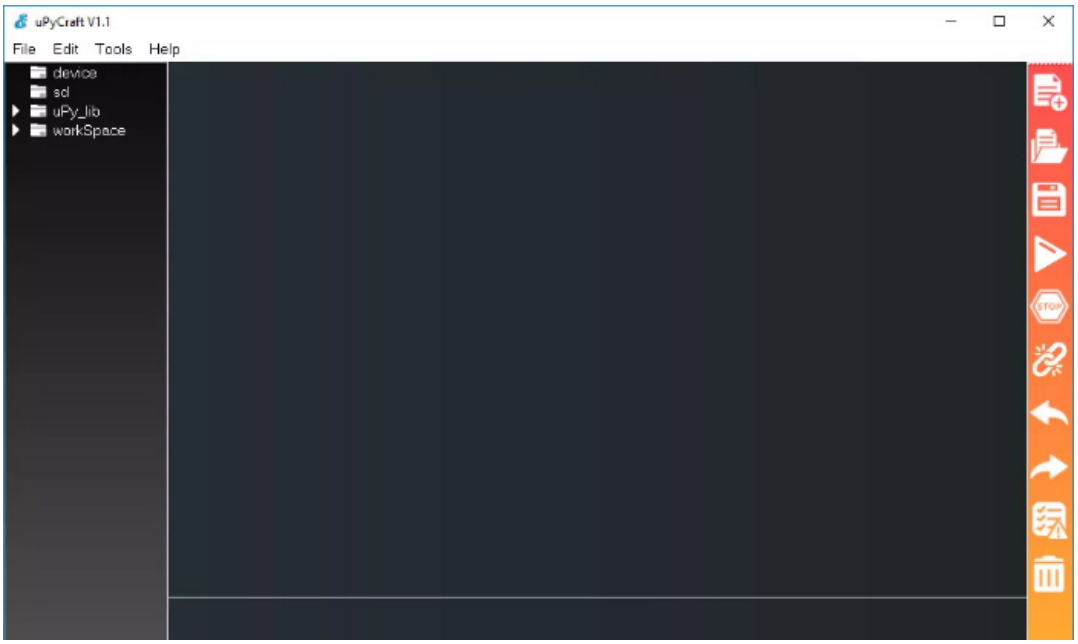
Δεξιά παρατηρούμε πως έχουμε τη δυνατότητα να ενεργοποιήσουμε και άλλα παράθυρα όπως π.χ. για την παρακολούθηση των τιμών των μεταβλητών (variables). Από το μενού «View» συνολικά μπορούμε να επιλέγουμε τα παράθυρα που θέλουμε να είναι ενεργά.



Αναλυτικότερα μπορούμε να δούμε πληροφορίες σε διάφορες σχετικές σελίδες όπως π.χ. την https://realpython.com/python-thonny

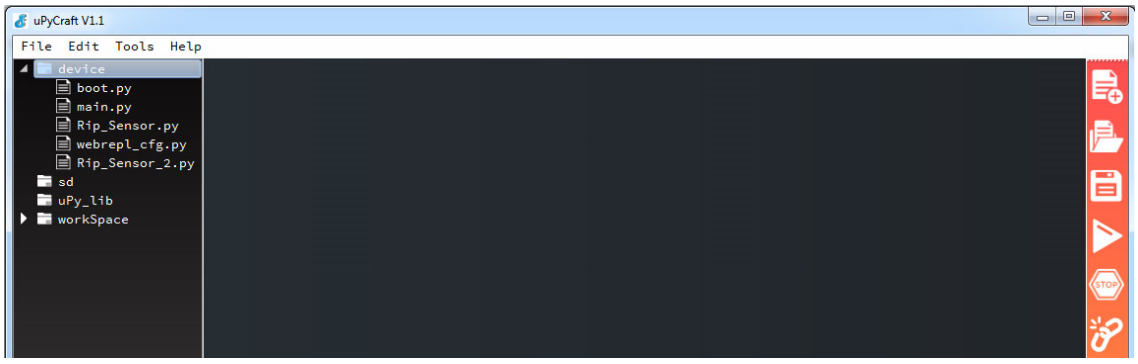
2.3 uPyGraft IDE

Αρχικά εγκαθιστούμε τον περιβάλλον ανάπτυξης Python "uPyGraft IDE", όπως περιγράφεται και στην ενότητα «7.1.3 - uPyCraft IDE». Μετά την ενεργοποίηση της εφαρμογής θα εμφανιστεί το ακόλουθο παράθυρο.



Ακολούθως συνδέουμε τον μικροελεγκτή μας. Εάν είναι η πρώτη φορά που τον χρησιμοποιούμε, ακολουθούμε τη διαδικασία που περιγράφεται στην ενότητα «7.1.4-Εγκατάσταση του Micropython Firmware» για να εγκαταστήσουμε στον μικροελεγκτή μας το κατάλληλο firmware και να μπορούμε πλέον να ξεκινήσουμε να τον προγραμματίζουμε.

Μόλις τελειώσει η διαδικασία εγκατάστασης πατάμε το εικονίδιο  στη δεξιά στήλη και συνδεόμαστε με τον μικροελεγκτή μας. Πλέον στην οθόνη μας έχουν γίνει κάποιες μικρές αλλαγές δηλαδή έχει εμφανιστεί στην αριστερή στήλη και ο φάκελος «device» στον οποίο βρίσκουμε τα αρχεία που ενδεχομένως περιέχονται σε αυτόν αλλά και το εικονίδιο  (σύνδεση με τον μικροελεγκτή) έχει μετατραπεί σε  (αποσύνδεση από τον μικροελεγκτή).



Επίσης ένας απλός τρόπος για να δούμε εάν έχει γίνει η σύνδεση του μικροελεγκτή μας με την εφαρμογή είναι και η δυνατότητα που έχουμε όπως και στην εφαρμογή Thonny IDE να γράψουμε στο παράθυρο Shell μια απλή εντολή όπως π.χ. print(“HALLO MY FRIEND”) και να δούμε εάν θα εκτελεστεί δηλαδή εάν θα εμφανιστεί το μήνυμα μας όπως ακολούθως:

```
>>> print("HALLO MY FRIEND")
HALLO MY FRIEND
>>>
```

Πλέον, όπως και στο περιβάλλον Thonny IDE που περιγράψαμε προηγουμένως, μπορούμε να γράψουμε ολόκληρο κώδικα στο επάνω δεξιά παράθυρο (editor) και να δούμε τα αποτελέσματα του στο κάτω δεξιά παράθυρο (Shell) πετώντας το εικονίδιο «Download And Run» () όπου πρώτα κατεβάζουμε τον κώδικα και τον αποθηκεύουμε στον μικροελεγκτή και μετά αρχίζει να εκτελείται.

2.4 ΓΛΩΣΣΕΣ ΑΝΑΠΤΥΞΗΣ ΛΟΓΙΣΜΙΚΟΥ

2.4.1 Προγραμματισμός σε περιβάλλον Arduino IDE

Μετά την εγκατάσταση του Arduino IDE μπορούμε να ξεκινήσουμε γράφοντας τα πρώτα μας τμήματα κώδικα (1). Το κείμενο που ακολουθεί περιγράφει τα βασικά σημεία του προγραμματισμού του Arduino και βασίζεται σε μεγάλο βαθμό στη βιβλιογραφική αναφορά (1), και παρέχεται για να προαχθεί η αυτονομία του κειμένου της εργασίας και να διευκολυνθεί ο αναγνώστης.

Η λογική του Arduino είναι πολύ απλή: στην ουσία υπάρχουν δύο βασικές συναρτήσεις, η **setup()** και η **loop()** οι οποίες δουλεύουν ως εξής:

- **setup()** - εδώ βάζουμε όλες τις εντολές που πρέπει να εκτελεστούν μία φορά, όταν ενεργοποιείται η μονάδα μας (όταν δηλαδή ξεκινά η παροχή ρεύματος στη συσκευή ή όταν πατηθεί το πλήκτρο reset που υπάρχει). Συνήθως εισάγονται αρχικοποιήσεις τιμών, μεταβλητών και οπωσδήποτε ο χαρακτηρισμός των εισόδων/εξόδων που θα χρησιμοποιήσουμε (αν δηλαδή ένα συγκεκριμένο Pin θα είναι είσοδος ή έξοδος).
- **loop()** - εδώ γράφουμε το πρόγραμμά. Οι εντολές που εισάγονται εδώ θα εκτελεστούν κι όταν ολοκληρωθεί η εκτέλεση του σώματος της loop, θα ξεκινήσει εκ νέου η εκτέλεσή της, συνεχίζοντας από την αρχή της σε έναν ατέρμονο βρόχο. Αυτός θα

επαναλαμβάνεται συνεχώς, όσο παρέχεται τροφοδοσία Arduino ή μέχρι να πατηθεί το πλήκτρο reset.

Στην περίπτωση του Reset ξανατρέχει η συνάρτηση setup() μία φορά και ακολούθως η loop() ξανά και ξανά, όπως δηλαδή ακριβώς και όταν αρχικά ενεργοποιείται με ρεύμα ο μικροελεγκτής. Στην περίπτωση που έχουμε κάνει αλλαγές στο πρόγραμμά μας και το φορτώσουμε στον μικροελεγκτή (η διαδικασία αυτή θα παρουσιαστεί παρακάτω) αρκεί να πατήσουμε το πλήκτρο Reset ώστε να φορτώσει το πρόγραμμά μας από την αρχή με τον τρόπο που περιγράφηκε.

```
void setup() {
  /* οι εντολές εδώ θα τρέξουν μόνο στην ενεργοποίηση ή μετά από Reset */
}
void loop() {
/* οι εντολές εδώ θα τρέχουν ξανά και ξανά,
μέχρι να απενεργοποιηθεί ή να πατηθεί το Reset */
}
```

2.4.1.1 Τύποι Μεταβλητών

Οι τύποι των μεταβλητών που υποστηρίζονται είναι αρκετοί, με συνηθέστερους τους:

- **boolean**, με τιμές το 0 και 1 (ή True – False)
- **byte**, με τιμές από 0 έως και 255
- **int**, ακέραιος με δυνατές τιμές από -32768 έως και 32767
- **long**, ακέραιος με δυνατές τιμές από -2147483648 έως και 2147483647
- **float**, δεκαδικοί αριθμοί
- **char**, ένας χαρακτήρας (μέγεθος ένα Byte)
- **string**, πίνακας χαρακτήρων

Ως παράδειγμα έχουμε:

```
int ledPin = 13; // ορίζω ακέραια μεταβλητή ledPin και αρχικοποιώ την τιμή της σε 13
float SinVal;   // ορίζω πραγματική μεταβλητή SinVal
```

2.4.1.2 Σχόλια

Για την τεκμηρίωση του κώδικα, προκειμένου για ευκολότερη κατανόηση και συντήρησή του, μπορούμε να εισάγουμε σχόλια. Μπορούμε να χρησιμοποιήσουμε τις δύο πλάγιες γραμμές (//)

για να τοποθετήσουμε σχόλια σε μία γραμμή ή τον συμβολισμό /*…….*/ για να εισάγουμε σχόλια περισσότερων γραμμών (ό,τι τοποθετήσουμε ανάμεσα στα /* και */ αγνοείται).

Παράδειγμα:

```
int ledPin = 13;      // ορίζω τον αριθμό του Pin για το LED
/* Στον κώδικα που ακολουθεί θα προγραμματίσουμε ένα
LED να αναβοσβήνει ανά 1 sec */
```

2.4.1.3 Χρήσιμες Συναρτήσεις και Δομές

- **Συνάρτηση Διαχείρισης Θυρών Εισόδου – Εξόδου (Pins)**

  Η συνάρτηση **pinMode(Pin, Mode)** καλείται παραθέτοντας το όνομά της και ορίσματα (α) τον αριθμό του Pin και (β) την κατάσταση λειτουργίας που χαρακτηρίζεται με τη λέξη INPUT (είσοδος) ή OUTPUT(έξοδος).

  Παράδειγμα:

```
pinMode(12, OUTPUT);
pinMode(ledPin, OUTPUT);
pinMode(A2, INPUT);
```

- **Ψηφιακή Έξοδος**

  Αυτό γίνεται με χρήση της συνάρτησης **digitalWrite(Pin, Value)**, όπου το όρισμα Pin αναφέρεται στο νούμερο της θύρας για την οποία θα δώσουμε τάση εξόδου, ενώ η τάση εξόδου μπορεί να είναι 0 V ή 5 V. Οι δύο αυτές τιμές αναπαρίστανται με προκαθορισμένες τιμές στην παράμετρο value

  - **LOW**: θα δώσει 0 V στην έξοδο (pin)
  - **HIGH**: θα δώσει 5 V στην έξοδο (pin)

  Για παράδειγμα:

```
digitalWrite(ledPin, HIGH);
```

  Προσοχή: Η αντίστοιχη θύρα θα πρέπει να έχει οριστεί ως εξόδου στη διαδικασία **setup()**, με χρήση της συνάρτησης **pinMode()**.

  Για παράδειγμα:

```
pinMode(10, OUTPUT);
```

- **Ψηφιακή Είσοδος**

Αυτό γίνεται με χρήση της συνάρτησης **digitalRead(Pin)**, όπου το όρισμα Pin αναφέρεται στο νούμερο της θύρας για την οποία θα πάρουμε είσοδο, ενώ η συνάρτηση επιστρέφει με το όνομά της την τιμή εισόδου. Η τάση εισόδου μπορεί να είναι 0V ή 5V, οι οποίες αναπαρίστανται με προκαθορισμένες τιμές στην τιμή που διαβάζουμε:

- **LOW**: όταν λάβει τάση 0 V στην είσοδο (pin)
- **HIGH**: όταν λάβει τάση 5 V στην είσοδο (pin)

Για παράδειγμα:

```
Val = digitalRead(ledPin);
```

Προσοχή: Η αντίστοιχη θύρα θα πρέπει να έχει οριστεί ως εισόδου στη διαδικασία **setup()**, με χρήση της συνάρτησης **pinMode()**.

Για παράδειγμα:

```
pinMode(10, INPUT);
```

- **Αναλογική Έξοδος**

Αυτό γίνεται με χρήση της συνάρτησης **analogWrite(Pin, Value)**, όπου το όρισμα Pin αναφέρεται στο νούμερο της θύρας για την οποία θα δώσουμε ρεύμα εξόδου, ενώ η τάση εξόδου κυμαίνεται από 0V μέχρι και 5V, οι οποίες τιμές της τάσης αναλογικά αναπαρίστανται με τιμές στη μεταβλητή value. Τιμή 0 δίνει 0V στην έξοδο (pin), τιμή 255 δίνει τάση 5V στην έξοδο (pin), ενώ αναλογικά μπορούμε να δώσουμε ενδιάμεσες τάσεις (π.χ. 122 για τάση 2,5V).

Για παράδειγμα:

```
analogWrite(ledPin, 122);
```

Προσοχή: Τη λειτουργία αυτή μπορούν να υποστηρίξουν μόνο τα PWM pins (συγκεκριμένα pins του Arduino που έχουν τη δυνατότητα προσομοίωσης αναλογικής εξόδου μέσω παλμοκωδικής διαμόρφωσης) κι όχι όλα τα ψηφιακά. Η αντίστοιχη θύρα θα πρέπει να έχει οριστεί ως εξόδου στη διαδικασία **setup()**, με χρήση της συνάρτησης **pinMode()**.

Για παράδειγμα:

```
pinMode(10, OUTPUT);
```

- **Αναλογική Είσοδος**

  Αυτό γίνεται με χρήση της συνάρτησης **analogRead(Pin)**, όπου το όρισμα Pin αναφέρεται στον αριθμό της θύρας για την οποία θα πάρουμε είσοδο, ενώ η συνάρτηση επιστρέφει ως αποτέλεσμα την τιμή που διαβάστηκε από τη θύρα. Η τιμή εισόδου κυμαίνεται από 0 μέχρι και 1023. Η θύρα πρέπει να είναι μία από τις θύρες του Arduino που επιτρέπουν αναλογική είσοδο (A0-A5). Συνήθως αποθηκεύουμε το αποτέλεσμα σε μια μεταβλητή.

  Για παράδειγμα:

```
int r = analogRead(A1);
```

  Προσοχή: Η αντίστοιχη θύρα θα πρέπει να έχει οριστεί ως εισόδου στη διαδικασία **setup()**, με χρήση της συνάρτησης **pinMode()**.

  Για παράδειγμα:

```
pinMode(A1, INPUT);
```

- **Συνάρτηση Καθυστέρησης**

  Συνήθως θέλουμε να εισάγουμε μια καθυστέρηση που θα διαρκέσει για το χρόνο που εμείς ορίζουμε. Αυτό το επιτυγχάνουμε με χρήση της συνάρτησης **delay(time)** όπου στη θέση time δίνουμε το χρόνο σε msec (1/1000 sec). Η εντολή delay(time) σημαίνει ότι σταματά στο σημείο αυτό η εκτέλεση του προγράμματός μας για χρόνο ίσο με *time*.

  Για παράδειγμα:

```
delay(1000); //σταματά την εκτέλεση του προγράμματος για 1000 ms = 1 sec
delay(500); //σταματά την εκτέλεση στο σημείο αυτό για 500 ms = 0.5 sec
```

- **Συνάρτηση Ενεργοποίησης Θύρας Επικοινωνίας**

  Για να ενεργοποιήσουμε τη σειριακή θύρα επικοινωνίας αρκεί να δώσουμε στη διαδικασία **setup()** την εντολή **Serial.begin(BaudRate)**, όπου το BaudRate εκφράζει το ρυθμό με τον οποίο θα μεταδίδονται τα bits (μια τιμή στα 9600 είναι συνήθως αρκετή).

  Για παράδειγμα:

```
Serial.begin(9600);
```

- **Δομή Επιλογής**

Στον προγραμματισμό πολλές φορές χρειάζεται να ελέγξουμε κάποια συνθήκη για να αποφασίσουμε αν θα εκτελεστεί ένα τμήμα κώδικα ή αν θα εκτελεστεί κάποιο άλλο αντί για αυτό στη θέση του.

Αυτό το επιτυγχάνουμε με χρήση της δομής επιλογής, η οποία συντάσσεται

```
if <συνθήκη>
   { <εντολές 1> }
else
   { <εντολές 2> }
```

όπου, στη <συνθήκη> παραθέτουμε τον έλεγχο που θέλουμε να γίνει, συνήθως χρησιμοποιώντας τους τελεστές σύγκρισης (>, <, =, <>, >=, <=), π.χ. time > 500.

Στα μπλοκ { <εντολές> } εκτελούνται αντίστοιχα οι εντολές που θέλουμε σε κάθε περίπτωση. Αν ισχύει η <συνθήκη> θα εκτελεστούν οι <εντολές 1>, αν δεν ισχύει οι <εντολές 2>. Σε κάθε περίπτωση, το τελευταίο κομμάτι else { <εντολές 2>} δεν είναι απαραίτητο να υπάρχει, στην οποία περίπτωση εάν η συνθήκη <συνθήκη> είναι ψευδής, δεν εκτελείται καθόλου κώδικας της δομής *if*.

- **Δομή Επανάληψης**

  Πολλές φορές χρειάζεται να επαναλάβουμε κάποια διαδικασία αρκετές φορές. Στην περίπτωση αυτή έχουμε εντολές οι οποίες επαναλαμβάνουν ένα σύνολο εντολών όσες φορές θέλουμε, είτε μετρώντας τις επαναλήψεις είτε ελέγχοντας κάθε φορά μία συνθήκη. Η συχνότερη μορφή που συναντάμε σε μια επανάληψη είναι αυτή με τον προκαθορισμένο αριθμό βημάτων. Η σύνταξη της εντολής αυτής είναι η εξής:

```
for (i=1;i<10;i=i+1) {
  brightness = brightness + 5;
  analogWrite(ledPin, brightness);
};
```

  Υπάρχουν εντολές επανάληψης που δεν έχουν προκαθορισμένο αριθμό βημάτων, αλλά συνεχίζουν επ' αόριστο ελέγχοντας μια συνθήκη.

  - while <συνθήκη> { <εντολές> } //**όσο ισχύει** η <συνθήκη> τρέχουν οι εντολές
  - repeat {<εντολές>} until <συνθήκη> // οι <εντολές> τρέχουν **όσο δεν ισχύει η συνθήκη**

2.4.2 MICROPYTHON

Η MicroPython (2), (3), (4) είναι μια περιορισμένη και αποδοτική υλοποίηση της γλώσσας προγραμματισμού Python 3, η οποία περιλαμβάνει ένα μικρό υποσύνολο της τυπικής βιβλιοθήκης Python και είναι βελτιστοποιημένη για να τρέχει σε μικροελεγκτές και σε περιβάλλοντα με περιορισμένους πόρους.

Ο προγραμματισμός στην MicroPython είναι πολύ παρόμοιος με τον προγραμματισμό στην Python: όλα τα γλωσσικά χαρακτηριστικά της Python περιλαμβάνονται και στην MicroPython, εκτός από μερικές εξαιρέσεις. Επειδή οι μικροελεγκτές και τα ενσωματωμένα συστήματα είναι πολύ πιο περιορισμένα από τους υπολογιστές μας, η MicroPython δεν διαθέτει ως προεπιλογή την πλήρη βιβλιοθήκη της Python.

2.4.2.1 Τύποι Μεταβλητών

Στη MicroPython κάθε μεταβλητή έχει έναν τύπο ανά πάσα στιγμή, αν και μπορεί να αλλάζει τον τύπο των δεδομένων που αποθηκεύει σε διαφορετικά σημεία εκτέλεσης του προγράμματος.

Οι κύριοι τύποι μεταβλητών της MicroPython είναι οι ακόλουθοι:

- **int**, ακέραιοι αριθμοί
- **float**, δεκαδικοί αριθμοί
- **str**, συμβολοσειρές
- **bool**, True ή False

2.4.2.2 Μαθηματικοί τελεστές και τελεστές σύγκρισης

Η MicroPython υποστηρίζει όλες τις τυπικές μαθηματικές πράξεις που συναντάμε σε όλες τις γλώσσες προγραμματισμού (+, -, *, /, % (ακέραιο υπόλοιπο διαίρεσης)), ενώ υποστηρίζει και τον τελεστή // που δίνει το ακέραιο πηλίκο της διαίρεσης δύο αριθμών. Η ανάθεση τιμής σε μεταβλητή γίνεται με τον τελεστή =, ενώ για τους τελεστές σύγκρισης υιοθετείται το αντίστοιχο σύνολο της C (==, !=, <, <=, >, >=).

2.4.2.3 Σχόλια

Στην MicroPython μπορούμε να εισάγουμε σχόλια χρησιμοποιώντας το σύμβολο της δίεσης (#).

Παράδειγμα:

```
x= a+b          #expect 3
```

2.4.2.4 Χρήσιμες Συναρτήσεις και Δομές

- **Συνάρτηση ελέγχου του τύπου μιας μεταβλητής**

  Η συνάρτηση καλείται με τη μορφή `type(μεταβλητή)` και επιστρέφει τον τύπο της μεταβλητής

Παράδειγμα:	a=3	b=3.5	c= “test”
	>>>type(a)	>>>type(b)	>>>type(c)
Επιστρέφει	<class ‘int’>	<class ‘float’>	<class ‘str’>

- **Δομή επιλογής**

  Η απλή δομή επιλογής συντάσσεται

```
if <συνθήκη>
    { <εντολές 1> }
else
    { <εντολές 2> }
```

  όπου στη <συνθήκη> έχουμε τον έλεγχο που θέλουμε να γίνει, συνήθως χρησιμοποιώντας τους τελεστές σύγκρισης (>, <, =, <>, >=, <=), π.χ. time > 500.

  Στα μπλοκ { <εντολές> } εκτελούνται αντίστοιχα οι εντολές που θέλουμε σε κάθε περίπτωση. Αν ισχύει η <συνθήκη> θα εκτελεστούν οι <εντολές 1>, αν δεν ισχύει οι <εντολές 2>. Σε κάθε περίπτωση, το τελευταίο κομμάτι else { <εντολές 2>} δεν είναι απαραίτητο να υπάρχει.

  Στη δομή επιλογής, αλλά και σε όλες τις δομές της Python, η στοίχιση των εντολών παίζει σημαντικό ρόλο: οι εντολές που πρέπει να εκτελεστούν εάν αληθεύει η *συνθήκη* πρέπει να τοποθετηθούν σε αυξημένη εσοχή σε σχέση με το *if*, και αντίστοιχα με τις εντολές που συσχετίζονται με το *else*. Κατόπιν, οι εντολές που ακολουθούν τη δομή *if*/*else* και πρέπει να εκτελεστούν ανεξάρτητα από το εάν αληθεύει η *συνθήκη* ή όχι, πρέπει να τοποθετηθούν στο ίδιο επίπεδο εσοχής με το *if*. Το ίδιο ισχύει και για όλες τις δομές που θα περιγραφούν στη συνέχεια.

- **Δομή Επανάληψης for**

  Η συχνότερη μορφή που συναντάμε σε μια επανάληψη είναι αυτή με τον προκαθορισμένο αριθμό βημάτων. Η σύνταξη της δομής αυτής είναι όπως το παρακάτω παράδειγμα:

```
number = 1
for number in range(1, 11):
        print(number)
```

- **Δομή Επανάληψης while**

  Στην περίπτωση που δεν έχουμε προκαθορισμένο αριθμό βημάτων χρησιμοποιούμε τη δομή επανάληψης **while**, η δομή της οποίας φαίνεται στο παρακάτω παράδειγμα:

```
number = 1
while number <= 10:
        print(number)
        number = number + 1
```

2.4.2.5 Έτοιμες συναρτήσεις και δομές για την πλακέτα ESP32

- **Για καθυστέρηση και χρονισμό**

  Χρησιμοποιώντας την ενότητα **time**

```
import time
time.sleep(1)          # sleep for 1 second
time.sleep_ms(500)     # sleep for 500 milliseconds
time.sleep_us(10)      # sleep for 10 microseconds
start = time.ticks_ms() # get millisecond counter
delta = time.ticks_diff(time.ticks_ms(), start) # compute time difference
```

- **Για τα Pins και είσοδο-έξοδο γενικού σκοπού**

  Η διαχείριση των pins καθώς και όλες οι λειτουργίες εισόδου-εξόδου γενικού σκοπού (general purpose I/O, GPIO) πραγματοποιούνται χρησιμοποιώντας την ενότητα βιβλιοθήκης **machine** και ειδικότερα τη μέθοδο **Pin** της ενότητας βιβλιοθήκης αυτής, όπως στα παραδείγματα που ακολουθούν:

```
from machine import Pin

p0 = Pin(0, Pin.OUT)    # create output pin on GPIO0
p0.on()                 # set pin to "on" (high) level
p0.off()                # set pin to "off" (low) level
p0.value(1)             # set pin to on/high

p2 = Pin(2, Pin.IN)     # create input pin on GPIO2
print(p2.value())       # get value, 0 or 1

p4 = Pin(4, Pin.IN, Pin.PULL_UP) # enable internal pull-up resistor
p5 = Pin(5, Pin.OUT, value=1) # set pin high on creation
```

- **Για δικτύωση**

  Όλες οι λειτουργίες δικτύωσης πραγματοποιούνται χρησιμοποιώντας την ενότητα βιβλιοθήκης **network**. Ειδικότερα, η ασύρματη δικτύωση υποστηρίζεται από την κλάση WLAN, όπως φαίνεται στο παρακάτω παράδειγμα.

```
import network

wlan = network.WLAN(network.STA_IF) # create station interface
wlan.active(True)       # activate the interface
wlan.scan()             # scan for access points
wlan.isconnected()      # check if the station is connected to an AP
wlan.connect('essid', 'password') # connect to an AP
wlan.config('mac')      # get the interface's MAC address
wlan.ifconfig()         # get the interface's IP/netmask/gw/DNS addresses

ap = network.WLAN(network.AP_IF) # create access-point interface
ap.config(essid='ESP-AP') # set the ESSID of the access point
ap.active(True)         # activate the interface
```

Μία χρήσιμη συνάρτηση για σύνδεση σε τοπικό δίκτυο είναι η ακόλουθη:

```
def do_connect():
        import network
        wlan = network.WLAN(network.STA_IF)
        wlan.active(True)
        if not wlan.isconnected():
        print ('connecting to network...')
        wlan.connect('essid', 'password')
        while not wlan.isconnected():
                pass
        print('network config:', wlan.ifconfig())
```

2.4.3 HTML

Η HTML (5), (6) έχει ονοματιστεί από τα αρχικά των λέξεων **H**yper **T**ext **M**arkup **L**anguage και είναι η δημοφιλέστερη γλώσσα για τη δημιουργία ιστοσελίδων. Δεν συγκαταλέγεται στις Γλώσσες Προγραμματισμού αλλά στις **γλώσσες σήμανσης** αφού χρησιμοποιεί σήμανση/«ετικέτες» (tags) για να περιγράψει τη δομή και το περιεχόμενο κάθε ιστοσελίδας.

Η γλώσσα HTML περιλαμβάνει ένα σύνολο ετικετών (**tags**) με την βοήθεια των οποίων περιγράφεται η δομή της ιστοσελίδας και η μορφοποίηση των στοιχείων της. Η γενική μορφή κάθε ετικέτας είναι: **<tag " ιδιότητες "> κείμενο… </tag>.** Οι ετικέτες εμφανίζονται κατά ζεύγη, η *ετικέτα ανοίγματος* και η *ετικέτα κλεισίματος*: όπως φαίνεται στη γενική μορφή, η ετικέτα κλεισίματος έχει το ίδιο όνομα ετικέτας με την ετικέτα ανοίγματος, αλλά πριν το όνομα της ετικέτας τοποθετείται ο χαρακτήρας «κάθετος» (slash - /). Υπάρχουν συγκεκριμένες ετικέτες, όπως η *meta* ή η *br* όπου στην ετικέτα ανοίγματος ενσωματώνεται και το κλείσιμο παραθέτοντας τον χαρακτήρα / *μετά* το όνομα της ετικέτας, π.χ. **
**. Καλό είναι οι ετικέτες να γράφονται με πεζά γράμματα.

Παράδειγμα:

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="UTF-8"/>
   <title> Μία απλή Ιστοσελίδα </title>
</head>

<body>
   <h1>Η Πρώτη μας Ιστοσελίδα!</h1>
```

```
    <p> Αυτή η ιστοσελίδα είναι εξαιρετικά απλή. Περιλαμβάνει μία
                        Επικεφαλίδα και μία παράγραφο.</p>
<!-- Εδώ μπορεί να γραφεί οποιοδήποτε σχόλιο το οποίο δεν θα εμφανιστεί
στην
        ιστοσελίδα. -->
</body>
</html>
```

Σχόλια στο Παράδειγμα

Γραμμή 1: Η ετικέτα *<!DOCTYPE html>* δηλώνει ότι οι εντολές που ακολουθούν είναι σε HTML5.

Γραμμή 4: Η ετικέτα *<meta charset="UTF-8"/>* δηλώνει την κωδικοποίηση του κειμένου. Σχεδόν πάντα πρέπει να χρησιμοποιείται η UTF-8.

Γραμμή 11: Τα σχόλια συνήθως μπαίνουν για να επεξηγήσουν τμήματα του κώδικα.

2.4.3.1 Elements

Έτσι ονομάζουμε μία ομάδα στοιχείων που απαρτίζεται από (α) την αρχή μιας ετικέτας (β) το κλείσιμο της και (γ) ό,τι περιλαμβάνεται μεταξύ της αρχής και του τέλους. Το ακόλουθο αποτελεί ένα element:

Παράδειγμα:

```
<h1>Η Πρώτη μας Ιστοσελίδα!</h1>
```

2.4.3.2 Ιδιότητες ετικετών (Attributes)

Οι ιδιότητες ετικετών βρίσκονται μέσα στην ετικέτα έναρξης ετικετών και δίνουν επιπλέον πληροφορίες/ιδιότητες για την ετικέτα. Οι τιμές των ιδιοτήτων πρέπει να βρίσκονται μέσα σε εισαγωγικά.

Παράδειγμα:

```
<h2><a id="top" >Κείμενο </a></h2>
```

Δύο εξαιρετικά χρήσιμες ιδιότητες είναι οι **id** και η **class**. Τις χρησιμοποιούμε όταν θέλουμε να θέλουμε να αντιστοιχίσουμε δικά μας ονόματα με elements, συνήθως για να αντιστοιχήσουμε σύνολα ιδιοτήτων με αυτά (π.χ για μορφοποίηση μέσω css) ή να αναφερθούμε σε αυτά μέσω γλωσσών προγραμματισμού, όπως η Javascript και η JQuery. Η τιμή της ιδιότητας **id** πρέπει να είναι μοναδική: όταν δύο ή περισσότερα elements θέλουμε να χρησιμοποιούν το ίδιο όνομα τότε

θα πρέπει η αντιστοίχιση να γίνει μέσω της **class**. Η ιδιότητα **class** μπορεί να προσδιορίζει συσχέτιση με περισσότερα από ένα ονόματα κλάσης.

Παράδειγμα:

```
<p class= "όνομα1 όνομα2" > Κείμενο … </p>
```

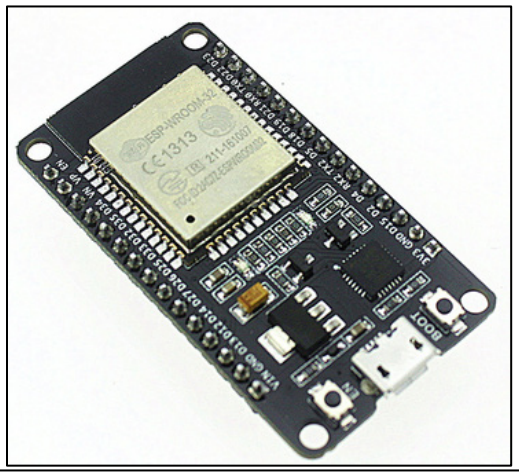
2.4.3.3 Βασικές Ετικέτες (Tags) της HTML

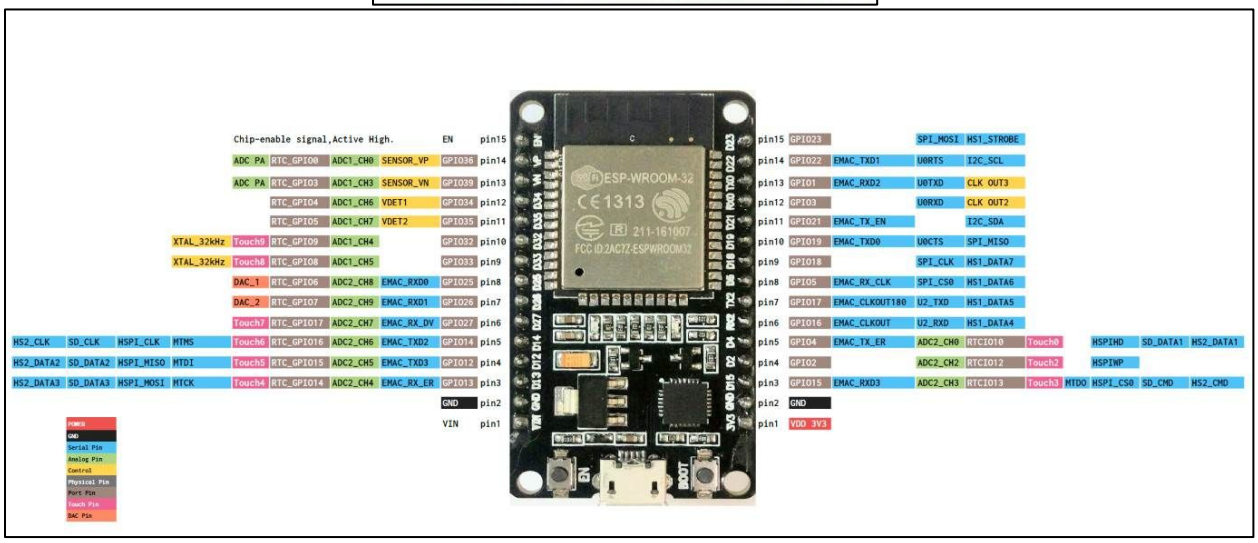
- **<html> ….. </html>,** Δηλώνει την αρχή και το τέλος μιας ιστοσελίδας
- **<head> ….. </head>,** Στο τμήμα αυτό της ιστοσελίδας περιλαμβάνονται πληροφορίες σχετικές με το περιεχόμενο της ιστοσελίδας
- **<body> ….. </body>,** Το ορατό τμήμα της ιστοσελίδας
- ** εντολές HTML ,** Ομαδοποιεί εντολές HTML προκειμένου να επηρεάσουμε την εμφάνιση τους με την βοήθεια CSS.
- **<title> ….. </title>,** Βρίσκεται μέσα στην ετικέτα **<head> </head>** και ορίζει τον τίτλο της ιστοσελίδας.
- **<hx> ….. </hx>,** Χρησιμοποιείται για τις επικεφαλίδες: **<h1>** για τις μεγαλύτερες επικεφαλίδες και **<h6>** για τις μικρότερες. Πριν και μετά από την επικεφαλίδα εμφανίζεται μία κενή γραμμή.
- ** ….. ,** Το κείμενο μέσα στην ετικέτα εμφανίζεται ως παράγραφος.
- ** ….. ,** Έντονη γραφή
- **<strike> …..</strike>,** Για να φαίνεται το κείμενο διαγραμμένο
- **
,** Αλλαγή γραμμής
- ** ,** Τα προγράμματα περιήγησης χρησιμοποιούν Για να προσθέσετε πραγματικά κενά στο κείμενό σας, μπορείτε να χρησιμοποιήσετε τον ειδικό χαρακτήρα ** **
- ** ….. ,** Για δημιουργία λίστας με κουκίδες. Συνδυάζεται με την ετικέτα **<li>**
- ** ….. ,** Για δημιουργία αριθμημένης λίστας. Συνδυάζεται με την ετικέτα **<li>**
- ** ….. ,** Καθορίζει τα στοιχεία μιας λίστας.
- ** …. ,** Το κείμενο ανάμεσα στο **<a>** και στο **** μετατρέπεται σε υπερσύνδεσμος που οδηγεί είτε σε άλλο σημείο της ίδιας ιστοσελίδας (δείτε το παράδειγμα που ακολουθεί) είτε σε άλλη ιστοσελίδα.
- **<table>…..</table>**, Δημιουργία πίνακα
- **<tr>…….</tr>**, ορίζει την γραμμή σε πίνακα
- **<td>…….</td>**, ορίζει τη στήλη σε πίνακα

- **<iframe src=** https://..... .gr **> ….. </iframe>,** Δημιουργεί ένα πλαίσιο και μέσα εμφανίζει μία ιστοσελίδα.
- **<form action= "……">……</form>,** Δημιουργία φόρμας για την εισαγωγή δεδομένων.

3. ΜΙΚΡΟΕΛΕΓΚΤΕΣ ΚΑΙ ΠΕΡΙΦΕΡΙΑΚΑ

3.1 ΜΙΚΡΟΕΛΕΓΚΤΗΣ ESP 32





Ο μικροελεγκτής ESP-32 (7), (8), (9), (10) είναι μονάδα ασύρματης σύνδεσης με εξαιρετικά χαμηλή κατανάλωση ενέργειας που βασίζεται στο chipset ESPRESSIF ESP32. Το ESP-32S ενσωματώνει επεξεργαστή διπλού πυρήνα, 448 KByte ROM, 520 KByte SRAM, 16 KByte SRAM σε RTC, 802.11 b / g / n / e / I Wi-Fi, Bluetooth v4.2 BR / EDR & BLE, άφθονες περιφερειακές διασυνδέσεις.

Ο μικροελεγκτής ESP-32 χρησιμοποιήθηκε στην κατασκευή των κυκλωμάτων για τον έλεγχο:

- Του Συστήματος της Πόρτας Γκαράζ
- Του Συστήματος του Φωτισμού Δωματίου

- Του Συστήματος του Κλιματισμού, καθώς και
- Του Συστήματος των Τιμών των Περιβαλλοντικών Μεταβλητών

Τα βασικά του χαρακτηριστικά που χρησιμοποιήθηκαν στην εργασία έχουν ως ακολούθως:

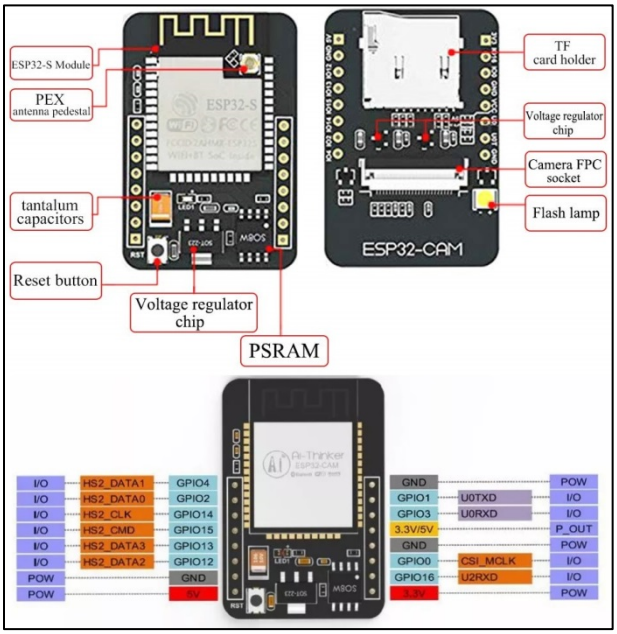
WiFi:

- 802.11 b/g/n/e/i
- 802.11 n（2.4 GHz）， up to 150 Mbps
- 802.11 e: QoS for wireless multimedia technology.
- WMM-PS, UAPSD
- MPDU and A-MSDU aggregation
- Block ACK
- Fragmentation and defragmentation
- Automatic Beacon monitoring/scanning
- 802.11 i security features: pre-authentication and TSN
- Wi-Fi Protected Access (WPA)/WPA2/WPA2-Enterprise/Wi-Fi Protected Setup (WPS)
- Infrastructure BSS Station mode/SoftAP mode
- Wi-Fi Direct (P2P), P2P Discovery, P2P Group Owner mode and P2P Power Management
- UMA compliant and certified
- Antenna diversity and selection

Peripheral Interfaces:

- 12-bit SAR ADC up to 18 channels
- 2 × 8-bit D/A converters
- 10 × touch sensors
- Temperature sensor
- 4 × SPI，2 × I2S，2 × I2C，3 × UART
- 1 host (SD/eMMC/SDIO)，1 slave (SDIO/SPI)
- Ethernet MAC interface with dedicated DMA and IEEE 1588 support
- CAN 2.0
- IR (TX/RX)
- Motor PWM，LED PWM up to 16 channels

3.2 ΜΙΚΡΟΕΛΕΓΚΤΗΣ ESP 32 CAM



Ο μικροελεγκτής ESP32-CAM βασίζεται στο ίδιο chip με τον ESP32 που αναφέραμε στην προηγούμενη υποενότητα και φέρει επιπλέον ενσωματωμένη κάμερα και υποδοχή για κάρτα μνήμης.

Στην παρούσα εργασία χρησιμοποιήθηκε στο Σύστημα Ελέγχου Χώρου με δύο ρόλους. Πρώτα ως φωτογραφική μηχανή, για να έχουμε στιγμιότυπα του χώρου κατά τη φάση της ενεργοποίησης και δεύτερον ως κάμερα για να έχουμε και συνεχή εικόνα μέσω διαδικτύου όποια στιγμή επιθυμούμε.

Τα βασικά του χαρακτηριστικά (11) (12) που χρησιμοποιήθηκαν στην εργασία έχουν ως ακολούθως:

- Low-power dual-core 32-bit CPU for application processors.
- Main frequency up to 240MHz, computing power up to 600 DMIPS.
- Built-in 520 KB SRAM, external 4M PSRAM.
- Supports interfaces such as UART/SPI/I2C/PWM/ADC/DAC.
- Support OV2640 and OV7670 cameras, built-in flash.
- Support image WiFi upload.
- Support TF card.
- Voltage: 5V

- Current: 2A
- Dimensions: 2.7x4cm

3.3 ΑΙΣΘΗΤΗΡΕΣ

Αισθητήρας ονομάζεται μία συσκευή που ανιχνεύει ένα φυσικό μέγεθος και παράγει από αυτό μία μετρήσιμη έξοδο.

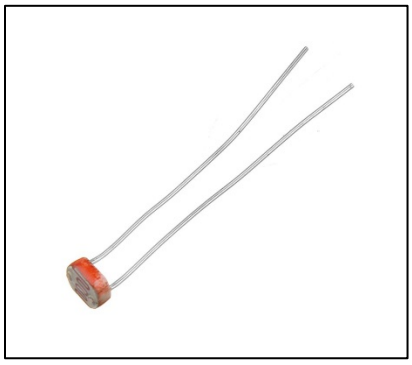
Έτσι συναντούμε αισθητήρες οι οποίοι:

- ανιχνεύουν την ύπαρξη φωτεινότητας και μεταφέρουν στον μικροελεγκτή (ή σε άλλες συνδεδεμένες διατάξεις) αντίστοιχη τιμή όπως τα φωτοστοιχεία,
- ανιχνεύουν την κίνηση και στέλνουν στον μικροελεγκτή αντίστοιχες τιμές, όπως ο HC-SR501,
- μπορούν με κατάλληλες συνδεσμολογίες να μας δώσουν τιμές για περιβαλλοντικές μεταβλητές π.χ. ατμοσφαιρική πίεση, θερμοκρασία περιβάλλοντος ή υγρασία ατμόσφαιρας όπως οι BMP-180, DHT-11 και DHT-22

3.3.1 GL5516 5mm Photoresistor LDR Light-Dependent Resistor

Η φωτοαντίσταση είναι μια μεταβλητή αντίσταση, η τιμή της οποίας τροποποιείται ανάλογα με το φως που πέφτει πάνω σε αυτή.

Χρησιμοποιήθηκε στην κατασκευή του Συστήματος Ελέγχου του Φωτισμού Δωματίου για να έχουμε ένδειξη μέσω διαδικτύου εάν είναι το φως ανοιχτό.



Χαρακτηριστικά (13)**:**

- Maximum Voltage: 150 Volt DC
- Maximum Wattage: 90mW

- Operating Temperature: -30 ~ +70 deg C
- Spectral Peak: 540nm
- Light Resistance (10 Lux): 5-10 Kohm
- Dark Resistance: 0.5 Mohm
- 100λ10: 0.5
- Response time: 30ms (Rise), 30ms (Down)
- Resistance Illumination: 2

3.3.2 PIR Motion Detector Module HC-SR501

Αισθητήρας ανίχνευσης κίνησης, συνήθως χρησιμοποιούμενος για ανίχνευση κίνησης ανθρώπων ή κατοικιδίων. Έχει τη δυνατότητα να ανιχνεύσει την κίνηση μέσα σε ένα δωμάτιο σε εμβέλεια έξι μέτρων.

Χρησιμοποιήθηκε στην κατασκευή του Συστήματος Ελέγχου Χώρου και για την ενεργοποίηση της διαδικασίας λήψης φωτογραφίας μόλις ανιχνεύσει κίνηση

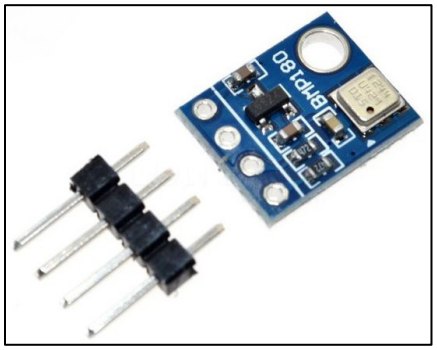
Χαρακτηριστικά (14)**:**

- Working voltage: DC5V to 20V
- Static consumption: 65 microamps
- Level output: 3.3V high, low 0V
- Time delay: Adjustable (0.3 seconds to 18 seconds)
- Blockade of the time: 0.2 seconds
- Trigger: L can not be duplicated, H can be repeated, the default value of H
- Sensing range: less than 120 degrees cone angle less than 7 m
- Working temperature: -15 to +70 degrees
- PCB Dimensions: 32 * 24mm ,screw pitch 28mm ,screw aperture 2mm, sensor lens Dimensions: (diameter): 23mm (default)

- Material: Mixture

3.3.3 BMP 180 Digital Barometric Pressure Sensor

Αισθητήρας καταγραφής της ατμοσφαιρικής πίεσης και της υγρασίας της ατμόσφαιρας. Χρησιμοποιήθηκε στην κατασκευή του Συστήματος Ελέγχου των αντίστοιχων Τιμών των Περιβαλλοντικών Μεταβλητών.



Χαρακτηριστικά (15)**,** (16)**:**

- Pressure range: 300 ... 1100hPa (+9000m... -500m relating to sea level)
- Supply voltage: 1.8 ... 3.6V (VDD)
  1.62V ... 3.6V (VDDIO)
- Low power: 5μA at 1 sample / sec. in standard mode
- Low noise: 0.06hPa (0.5m) in ultra low power mode
  0.02hPa (0.17m) advanced resolution mode
- I2C interface
- Fully calibrated
- Pb-free, halogen-free and RoHS compliant,
- MSL 1

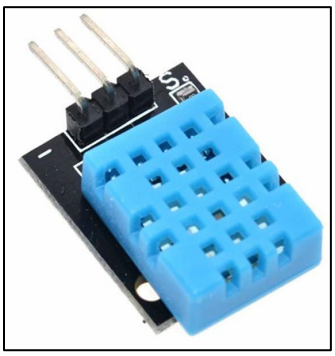
Εφαρμογές:

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

3.3.4 DHT-11 Digital Temperature and Humidity Sensor

Αισθητήρας καταγραφής της θερμοκρασίας του περιβάλλοντος και της υγρασίας της ατμόσφαιρας.

Χρησιμοποιήθηκε στην κατασκευή του Συστήματος Κλιματισμού για την παροχή της τιμής της θερμοκρασίας και της υγρασίας του δωματίου.

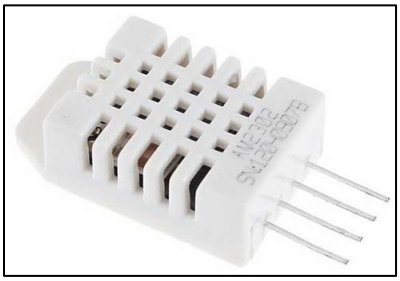


Χαρακτηριστικά (17)**:**

- Supply voltage: 3.3 ~ 5.5V DC
- Output: single-bus digital signal
- Measuring range: Humidity 20-90% RH, Temperature 0 ~ 50 ℃
- Accuracy: Humidity + -5% RH, temperature + -2 ℃
- Resolution: Humidity 1% RH, temperature 1 ℃

3.3.5 DHT-22 / AM2302 Digital Temperature and Humidity Sensor

Αισθητήρας καταγραφής της θερμοκρασίας του περιβάλλοντος και της υγρασίας της ατμόσφαιρας. Χρησιμοποιήθηκε στην κατασκευή του Συστήματος Ελέγχου των Τιμών των Περιβαλλοντικών Μεταβλητών για την παροχή της τιμής της θερμοκρασίας της ατμόσφαιρας.

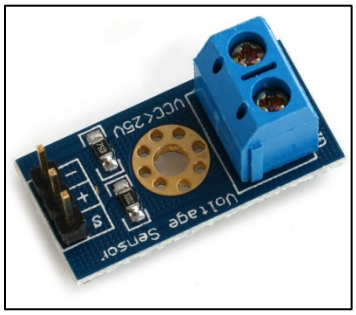


Χαρακτηριστικά (18)**:**

- Type:AM2302.
- Accuracy resolution:0.1.
- Temperature range:-40~80℃.
- Temperature measurement precision:±0.5℃.
- 4-pin package.
- Ultra-low power.
- Excellent long-term stability.
- All calibration, digital output.
- Completely interchangeable.
- Long distance signal transmission.
- Relative humidity and temperature measurement.

3.3.6 Voltage Detection Module – Voltage Sensor

Είναι αισθητήρας ανίχνευσης και καταμέτρησης της τάσης ή αλλιώς της διαφοράς δυναμικού μεταξύ δύο σημείων ενός ηλεκτρικού κυκλώματος και χρησιμοποιήθηκε στην κατασκευή του Συστήματος Κλιματισμού για να μας παρέχει ένδειξη μέσω διαδικτύου για την κατάσταση (ανοιχτός ή κλειστός) του κλιματισμού.



Χαρακτηριστικά (19)**:**

- Voltage input range: DC0-25 V
- Voltage detection range: DC0.02445 V-25 V
- Voltage analog resolution: 0.00489 V
- Output interface: "+ " connected 5/3.3V, "-" connected GND, "s" connected AD pins
- DC input interface: red terminal positive with VCC, negative with GND

3.3.7 Relays

Ο ηλεκτρονόμος, relay ή ρελέ είναι ένας ηλεκτρικός διακόπτης που ανοίγει και κλείνει ένα ηλεκτρικό κύκλωμα κάτω από τον έλεγχο ενός άλλου ηλεκτρικού κυκλώματος. Επειδή ένας ηλεκτρονόμος είναι ικανός να ελέγχει ένα κύκλωμα εξόδου υψηλότερης ισχύος από το κύκλωμα εισόδου, μπορεί να θεωρηθεί, γενικά, μια μορφή ηλεκτρικού ενισχυτή.

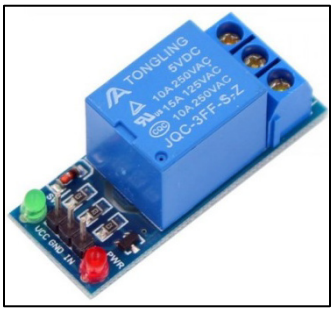
Κάθε επαφή ενός ηλεκτρονόμου μπορεί να είναι Κανονικά-Ανοικτή (Normally Open, NO), Κανονικά-Κλειστή (Normally Closed, NC) ή μεταγωγικός (change-over), ανάλογα με τον τύπο της.

Οι δύο πρώτοι τύποι ρελέ χρησιμοποιήθηκαν στην κατασκευή των Συστημάτων Ελέγχου

- Του Κλιματισμού
- Του Φωτισμού Δωματίου

και ο τρίτος τύπος ρελέ χρησιμοποιήθηκε στην κατασκευή το Συστήματος Ελέγχου της Πόρτας Γκαράζ

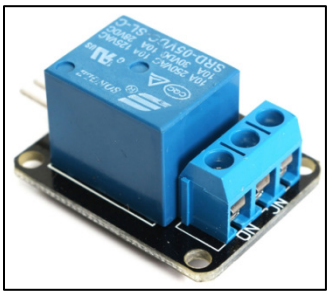
3.3.7.1 5V 1-Channel Relay Module Board



Βασικά Χαρακτηριστικά (20):

- 5V 1-Channel Relay interface board, and each one needs 15-20mA Driver Current
- Equipped with high-current relay, AC250V 10A - DC30V 10A
- Standard interface that can be controlled directly by microcontroller, such as Arduino, 8051, AVR, PIC, DSP, ARM and so on
- Contact independent wiring, safe and reliable
- Size:4.7 x 2.9 x 1.8cm

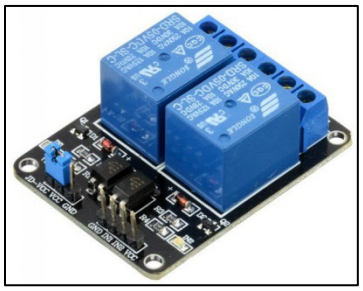
3.3.7.2 5V 1-Channel Relay Module Board



Βασικά Χαρακτηριστικά (21)**:**

- Trigger mode: high level
- 5V relay module
- Can be used for microcontroller development board module or home appliance control
- 5V - 12V TTL control signal
- Can control DC or AC signal, also 220V AC load
- Normally open / closed contact
- Power indicator
- With screw holes for easy installation
- Increase the transistor drive relay coils, control pins high impedance
- Control pins have pull down circuit to prevent malfunction of relay vacant

3.3.7.3 5V 2-Channel Relay Module Board



Βασικά Χαρακτηριστικά (22)**:**

- 5V 2-Channel Relay interface board, and each one needs 15-20mA Driver Current
- Equipped with high-current relay, AC250V 10A; DC30V 10A
- Standard interface that can be controlled directly by microcontroller (Arduino , 8051, AVR, PIC, DSP, ARM, ARM, MSP430, TTL logic)

3.3.8 Διακόπτες

3.3.8.1 Μαγνητική Επαφή

Χρησιμοποιήθηκαν δύο σετ στο Σύστημα Ελέγχου της Πόρτας του Γκαράζ με δύο χρήσεις:

- Ως διακόπτες για να σταματά η κίνηση του μοτέρ της πόρτας όταν αυτή φτάνει στα δύο ακραία όρια της επιτρεπόμενης κίνησης της, αλλά και
- Για να έχουμε ένδειξη μέσω διαδικτύου για την κατάσταση της πόρτας δηλαδή εάν είναι ανοιχτή.

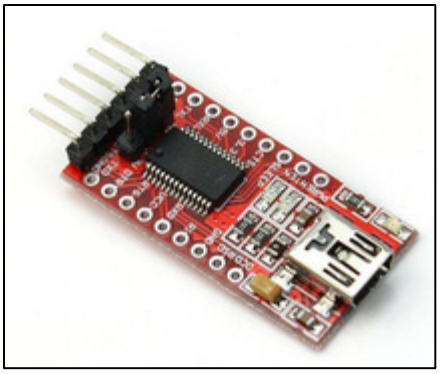
3.3.8.2 Διακόπτης ON-OFF

Χρησιμοποιήθηκε στα Συστήματα Ελέγχου του Φωτισμού Δωματίου και Κλιματισμού ως χειροκίνητη επιλογή της κατάστασης του φωτισμού και του κλιματισμού αντίστοιχα.

3.3.9 Λοιπά Περιφερειακά

3.3.9.1 FTDI FT232RL USB to TTL Serial Converter Adapter Module 5V and 3.3V

Με τον προσαρμογέα αυτόν έχουμε τη δυνατότητα σύνδεσης του μικροελεγκτή ESP32-CAM στον υπολογιστή μας, μέσω θύρας USB. Χρησιμοποιήθηκε στο Σύστημα Ελέγχου Χώρου για τον προγραμματισμού του, καθώς και την παροχή ρεύματος σε αυτόν.

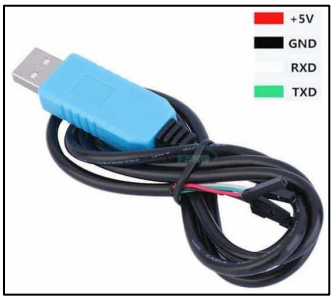


Μερικά από τα χαρακτηριστικά (23) του είναι:

- The FT232R is a USB to serial UART interface
- A low-cost way to add USB capability to For Arduino or other microcontrollers. Use this to give your own breadboard For Arduino USB capability for boot loading or downloading sketches.
- RXD/TXD transceiver communication indicator
- Color: Red
- USB Power Supply: 3.3V or 5V
- Size:43*17mm
- Net weight: 4 g

3.3.9.2 PL2303TA USB TTL to RS232 Converter Serial Cable

Η χρήση του καλωδίου PL2303TA είναι παρόμοια με αυτή του προσαρμογέα FT232RL δηλαδή και με αυτό έχουμε τη δυνατότητα σύνδεσης του μικροελεγκτή ESP32-CAM μέσω USB με τον υπολογιστή μας. Χρησιμοποιήθηκε στο Σύστημα Ελέγχου Χώρου για τον προγραμματισμού του, καθώς και την παροχή ρεύματος σε αυτόν.



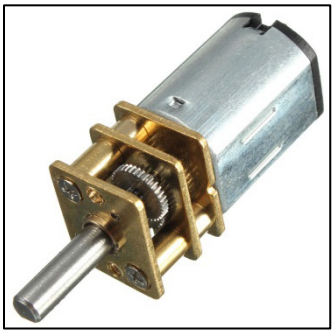
Μερικά από τα κύρια χαρακτηριστικά (24) του είναι:

- Single-chip upgrade solution for Legacy RS232 devices to USB interface

- USB to RS232 converters/cables/dongles
- This product is designed for laboratory, product testing, low-cost MCU communications and other applications, there are four lead,
  - Red +5 V
  - Black GND
  - White RXD
  - Green TXD

3.3.9.3 6V DC Metal Mini Motor 30RPM

Είναι ένα μοτέρ μεταλλικού περιβλήματος και γραναζώματος, σε κατάσταση κανονικής λειτουργίας πρέπει να τροφοδοτείται με συνεχή τάση 6V όπου στην κατάσταση αυτή πραγματοποιεί 30 περιστροφές ανά λεπτό. Χρησιμοποιήθηκε στην κατασκευή του Συστήματος Ελέγχου της Πόρτας του Γκαράζ.



Μερικά από τα χαρακτηριστικά (25) του είναι:

- Material:Metal
- Color:Silver
- Model:GA12-N20
- Total Size:33 x 12 x 9mm/1.30 x 0.47 x 0.35""(HxLxW)
- Working voltage range:DC1.5- 12.0V
- Nominal voltage:DC6V
- Rotational speed:30rpm
- Weight:9g

3.3.9.4 Αντίσταση 1kΩ

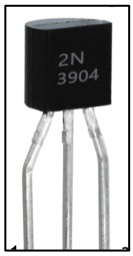
Χρησιμοποιήθηκε στην κατασκευή του κυκλώματος στο Σύστημα Ελέγχου Χώρου

3.3.9.5 Αντίσταση 10kΩ

Χρησιμοποιήθηκε στην κατασκευή του κυκλώματος στο Σύστημα Ελέγχου Χώρου

3.3.9.6 Τρανζίστορ 2N3904

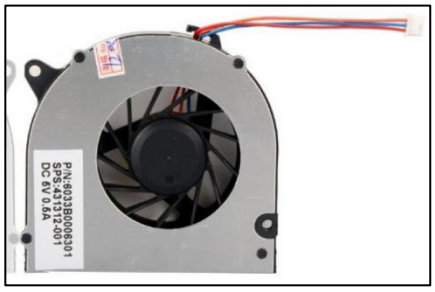
Χρησιμοποιήθηκε στην κατασκευή του κυκλώματος στο Σύστημα Συναγερμού



(26)

3.3.9.7 Ανεμιστήρας για Laptop

Ανεμιστήρας συνεχούς τάσης 6V που χρησιμοποιήθηκε στην κατασκευή του κυκλώματος στο Σύστημα Κλιματισμού



4. ΠΕΡΙΓΡΑΦΗ ΥΠΟΣΥΣΤΗΜΑΤΩΝ

Σε αυτή την ενότητα θα παρουσιάσουμε τα υποσυστήματα που δημιουργήθηκαν και αποτελούν αυτόνομα συστήματα για τη δημιουργία ενός «Έξυπνου Σπιτιού» - «Smart Home» με γνώμονα το χαμηλό κόστος αλλά ταυτόχρονα την αξιοπιστία και την ευκολία στη χρήση αλλά και την συντήρηση του.

4.1 ΚΥΚΛΩΜΑ ΕΛΕΓΧΟΥ ΠΟΡΤΑΣ ΓΚΑΡΑΖ

4.1.1 Περιγραφή

Το κύκλωμα Ελέγχου Πόρτας Γκαράζ αποτελείται από:

- 1 Μικροελεγκτή ESP32
- 1 Ρελέ 2-καναλιών
- 2 Μαγνητικές Επαφές
- 1 Μοτέρ
- 1 Λαμπάκι 12V
- Πηγή Τάσης 5V
- Πηγή Τάσης 12V

Ο προγραμματισμός του έγινε με χρήση της MicroPython (27) και αμέσως μετά παρουσιάζονται:

- Η σχηματική αναπαράσταση του κυκλώματος που κατασκευάστηκε
- Εικόνες από την κατασκευή του συστήματος
- Εικόνα από το web interface της ιστοσελίδας ελέγχου του συστήματος
- Η Οικονομική Ανάλυση του υποσυστήματος

Στο παράρτημα παρατίθενται και τα προγράμματα οδήγησης.

Η λειτουργία του συστήματος ελέγχεται μέσω του web interface στο οποίο παρέχεται η πληροφορία για την κατάσταση στην οποία βρίσκεται η πόρτα δηλαδή εάν είναι ανοιχτή ή κλειστή. Η πληροφορία αυτή παρέχεται από την τιμή που λαμβάνει ο μικροελεγκτής από την μαγνητική επαφή No1. Με βάση την ένδειξη έχουμε την επιλογή:

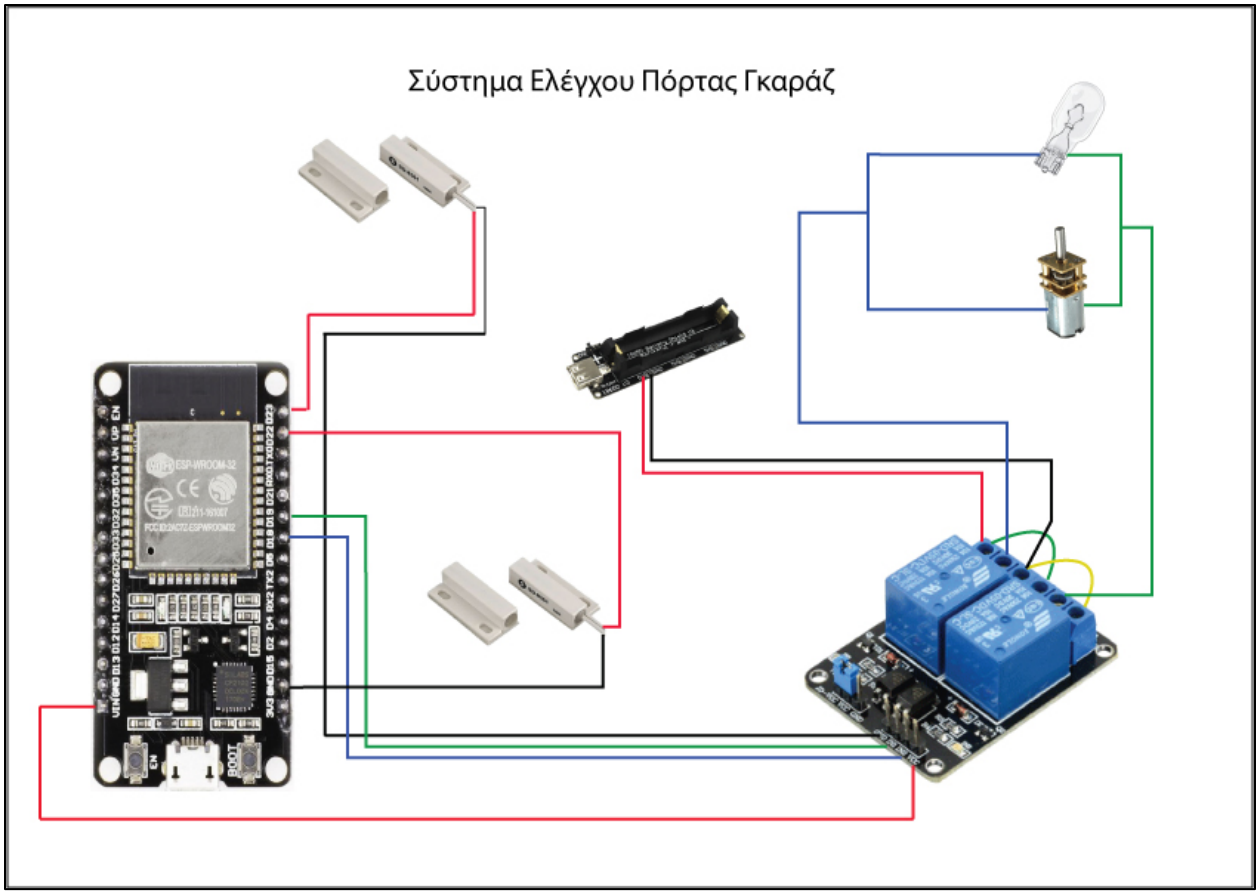
- Να ανοίξουμε την πόρτα πατώντας το πλήκτρο 

,

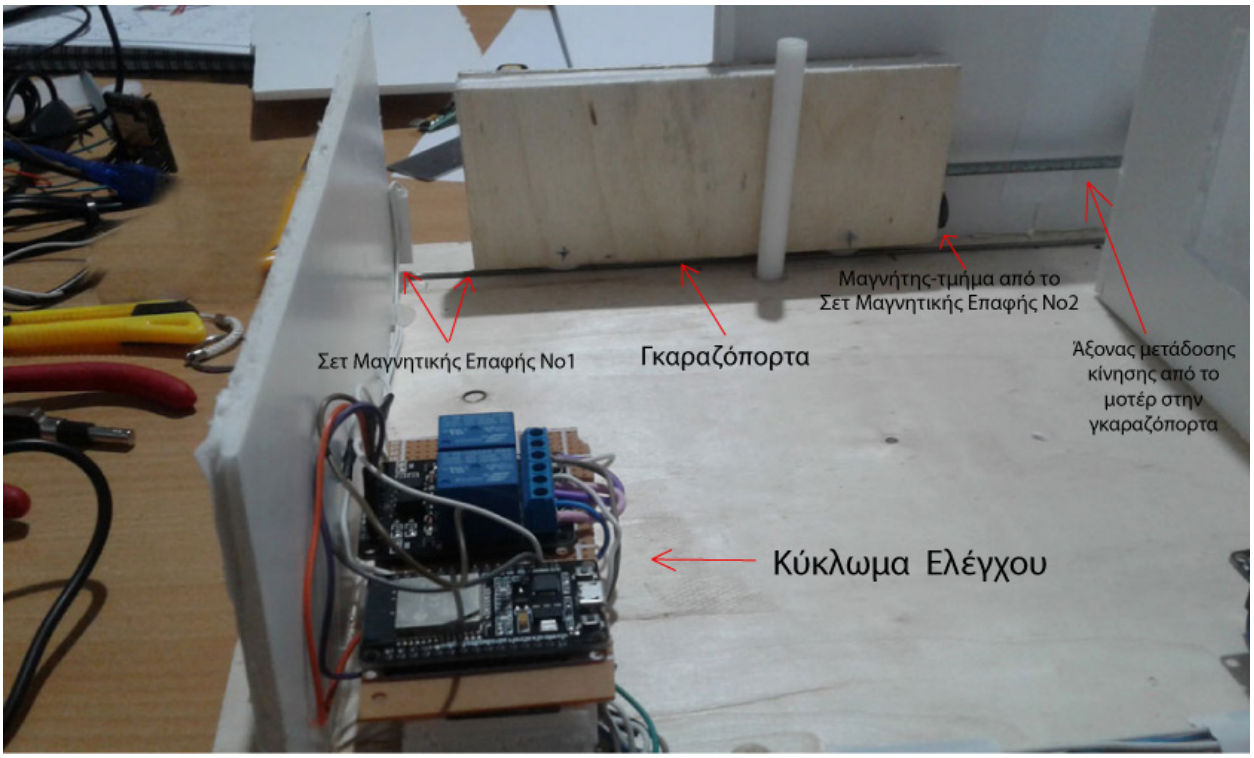
- Να κλείσουμε την πόρτα πατώντας το πλήκτρο CLOSE,
- Να σταματήσουμε την κίνηση της πατώντας το πλήκτρο STOP

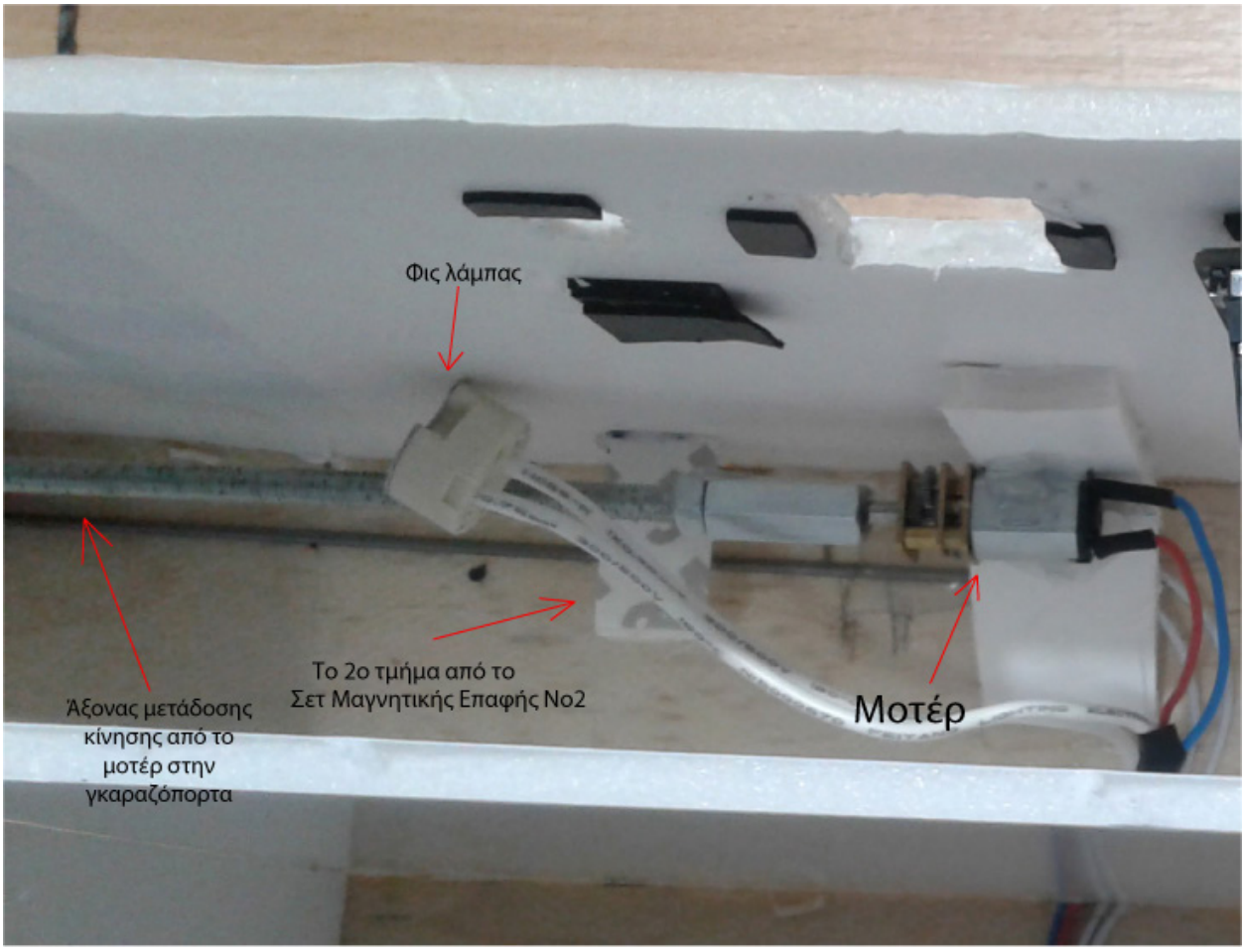
Η χρήση της Μαγνητικής επαφής No1 εκτός από την παροχή πληροφορίας για την κατάσταση στην οποία βρίσκεται η πόρτα, είναι και ως διακόπτης για να σταματήσει η κίνηση της όταν φτάσει στο κατάλληλο σημείο όπου είναι κλειστή. Από την άλλη πλευρά, η χρήση της Μαγνητικής επαφής No2 είναι ότι λειτουργεί ως διακόπτης για να σταματήσει η κίνηση της όταν φτάσει στο κατάλληλο σημείο όπου είναι τελείως ανοιχτεί.

Αξίζει να σημειωθεί πως το ίδιο κύκλωμα μπορεί να χρησιμοποιηθεί και για αυτοματισμούς σε πέργκολες, ηλεκτρικά παράθυρα – εξώφυλλα, ηλεκτρικές τέντες κ.τ.λ. με μικρές αλλαγές.



Σχήμα 1. Κύκλωμα συστήματος ελέγχου πόρτας γκαράζ


Σετ Μαγνητικής Επαφής No1
Γκαραζόπορτα
Μαγνήτης-τμήμα από το
Σετ Μαγνητικής Επαφής No2
Άξονας μετάδοσης
κίνησης από το
μοτέρ στην
γκαραζόπορτα
Κύκλωμα Ελέγχου


Φις λάμπας
Άξονας μετάδοσης
κίνησης από το
μοτέρ στην
γκαραζόπορτα
Το 2ο τμήμα από το
Σετ Μαγνητικής Επαφής No2
Μοτέρ

4.1.2 Οικονομική Ανάλυση

Ακολουθεί ο πίνακας που παραθέτει το κόστος αγοράς των υλικών που απαιτήθηκαν για την κατασκευή του κυκλώματος ελέγχου της πόρτας του γκαράζ.

Πίνακας οικονομικής ανάλυσης 1.: Οικονομική ανάλυση του κυκλώματος ελέγχου της πόρτας του γκαράζ

Α/Α	ΠΕΡΙΓΡΑΦΗ	ΠΟΣΟΤΗΤΑ	ΚΟΣΤΟΣ
1	Espessif ESP 32	1	13,5 €
2	Relay 2-Channel 5V	1	4,4 €
3	Μαγνητικές Επαφές	2	4,0 €
4	6V DC Metal Mini Motor	1	6,0 €
5	Λαμπάκι 12V	1	1,5 €
6	Πηγή Τάσης 5V	1	6,0 €
7	Πηγή Τάσης 12V	1	3,0 €
		ΣΥΝΟΛΟ	38,4 €

4.2 ΚΥΚΛΩΜΑ ΕΛΕΓΧΟΥ ΦΩΤΙΣΜΟΥ ΔΩΜΑΤΙΟΥ

4.2.1 Περιγραφή

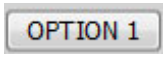
Το κύκλωμα Ελέγχου Φωτισμού Δωματίου αποτελείται από:

- 1 Μικροελεγκτή ESP32
- 1 Ρελέ 1-καναλιού
- 1 Ρελέ 1-καναλιού
- 1 Photoresistor
- 1 Διακόπτης 2 επαφών
- 1 Λαμπάκι 12V
- Πηγή Τάσης 5V
- Πηγή Τάσης 9V

Ο προγραμματισμός του έγινε με χρήση της MicroPython (27) και αμέσως μετά παρουσιάζονται:

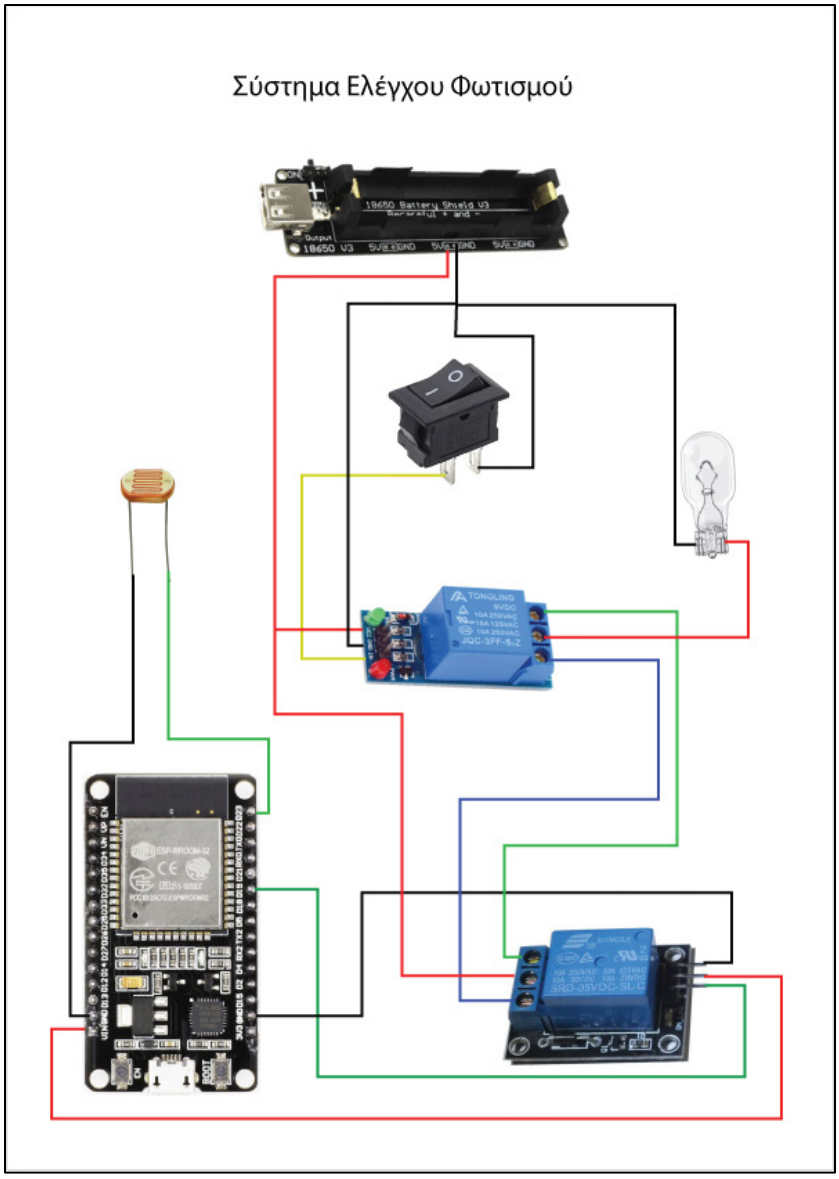
- Η σχηματική αναπαράσταση του κυκλώματος που κατασκευάστηκε
- Εικόνες από την κατασκευή του συστήματος
- Εικόνα από το web interface της ιστοσελίδας ελέγχου του συστήματος
- Η Οικονομική Ανάλυση του υποσυστήματος

Στα παραρτήματα παρατίθενται και τα προγράμματα οδήγησης.

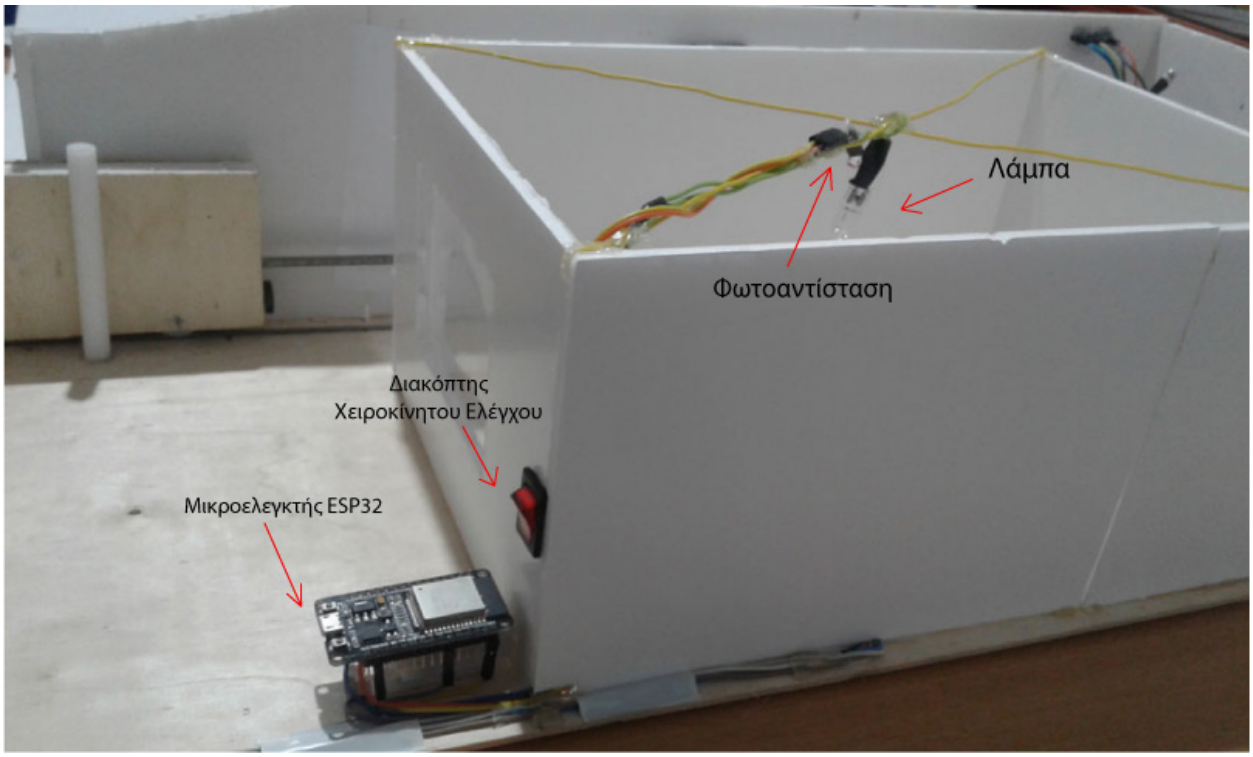
Η λειτουργία του συστήματος ελέγχεται μέσω του web interface στο οποίο παρέχεται και η πληροφορία για την κατάσταση στην οποία βρίσκεται ο φωτισμός δηλαδή εάν η λάμπα είναι ανοιχτή ή κλειστή. Η πληροφορία αυτή παρέχεται από την τιμή που λαμβάνει ο μικροελεγκτής από την φωτοαντίσταση. Με βάση την ένδειξη έχουμε την επιλογή να ανοίξουμε ή να κλείσουμε τον φωτισμό πατώντας τα κουμπιά 

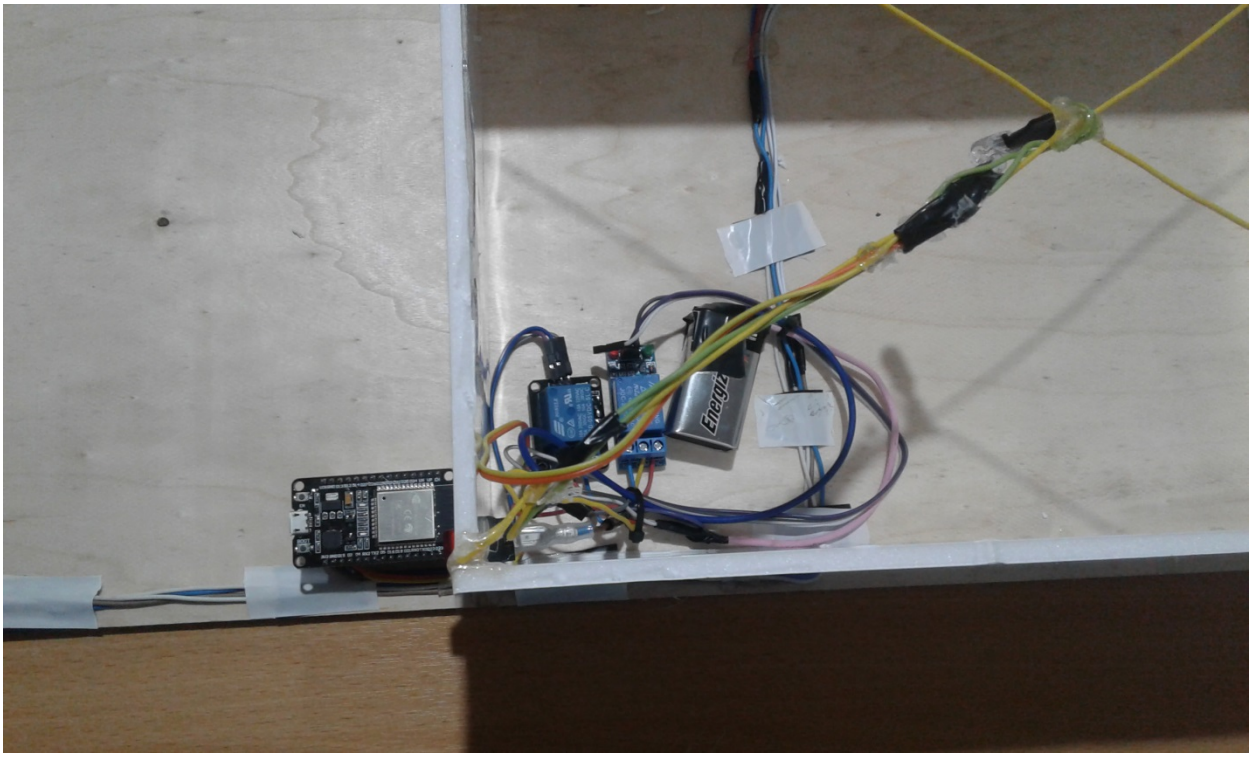
ή 

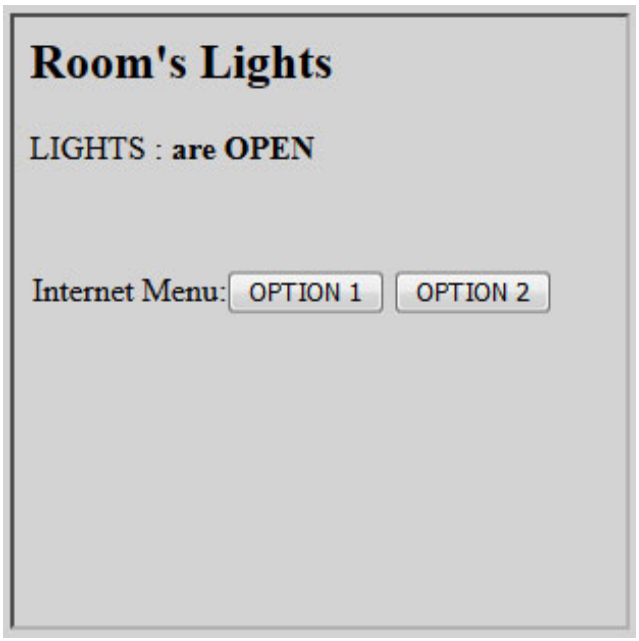
ανάλογα πάντα με την ένδειξη της κατάστασης του φωτισμού. Την κατάσταση του φωτισμού επίσης έχουμε τη δυνατότητα να την ελέγξουμε και χειροκίνητα μέσω του διακόπτη.



Σχήμα 2. Κύκλωμα συστήματος ελέγχου φωτισμού


Λάμπα
Φωτοαντίσταση
Διακόπτης
Χειροκίνητου Ελέγχου
Μικροελεγκτής ESP32


Energizer



4.2.2 Οικονομική Ανάλυση

Ακολουθεί ο πίνακας που δείχνει το κόστος αγοράς των υλικών που απαιτήθηκαν για την κατασκευή του κυκλώματος ελέγχου του φωτισμού δωματίου το οποίο μπορεί να χρησιμοποιηθεί και για τον έλεγχο κατάστασης – λειτουργίας και άλλων ηλεκτρικών συσκευών όπως θερμοσίφωνα, ηλεκτρική κουζίνα κ.τ.λ.

Πίνακας οικονομικής ανάλυσης 2. Οικονομική ανάλυση του κυκλώματος ελέγχου φωτισμού

Α/Α	ΠΕΡΙΓΡΑΦΗ	ΠΟΣΟΤΗΤΑ	ΚΟΣΤΟΣ
1	Espessif ESP 32	1	13,5 €
2	Relay 1-Channel 5V	1	2,5 €
3	Relay 1-Channel 5V	2	2,0 €
4	GL5516 5mm Photoresistor	1	0,2 €
5	Rocker Switch 2pin On-Off	1	0,5 €
6	Λαμπάκι 12V	1	1,5 €
7	Πηγή Τάσης 5V	1	6,0 €
8	Πηγή Τάσης 9V	1	3,0 €
		ΣΥΝΟΛΟ:	29,2 €

4.3 ΚΥΚΛΩΜΑ ΕΛΕΓΧΟΥ ΧΩΡΟΥ

4.3.1 Περιγραφή

Το κύκλωμα Ελέγχου Χώρου αποτελείται από:

- 2 Μικροελεγκτή ESP32 CAM
- 1 Αντίσταση 1kΩ
- 1 Αντίσταση 10kΩ
- 1 Τρανζίστορ 2N3904
- 1 PIR Sensor
- 1 Καλώδιο PL2303TA
- 1 Προσαρμογέα FTDI FT232RL
- 2 Πηγές Τάσης 5V

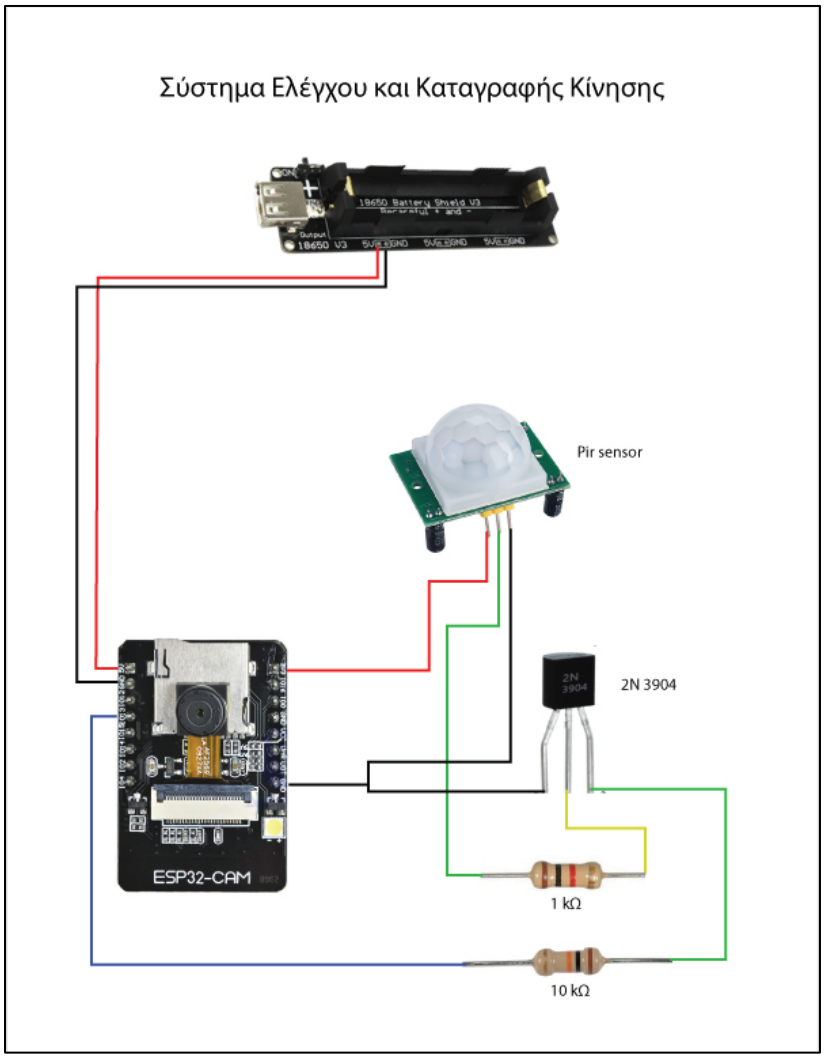
Ο προγραμματισμός του έγινε με χρήση του Arduino IDE. Ο ένας μικροελεγκτής προγραμματίστηκε (28) για να έχουμε εικόνα-βίντεο του χώρου και ο δεύτερος είναι τμήμα του υποσυστήματος για να έχουμε και φωτογραφίες του χώρου όταν παρατηρηθεί κίνηση από τον σένσορα (29).

Αμέσως μετά παρουσιάζονται:

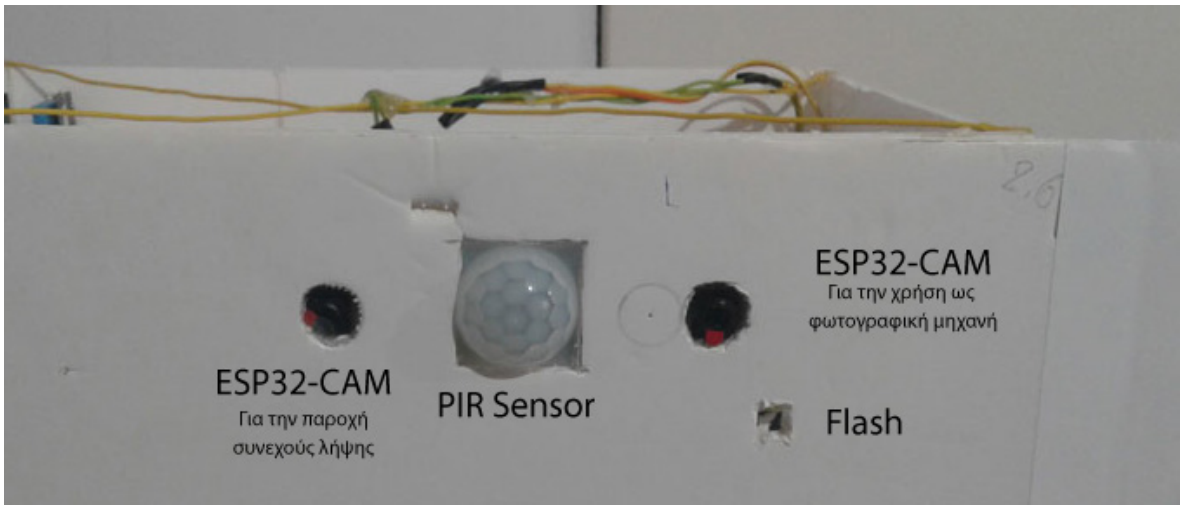
- Η σχηματική αναπαράσταση του κυκλώματος που κατασκευάστηκε
- Η Οικονομική Ανάλυση του υποσυστήματος

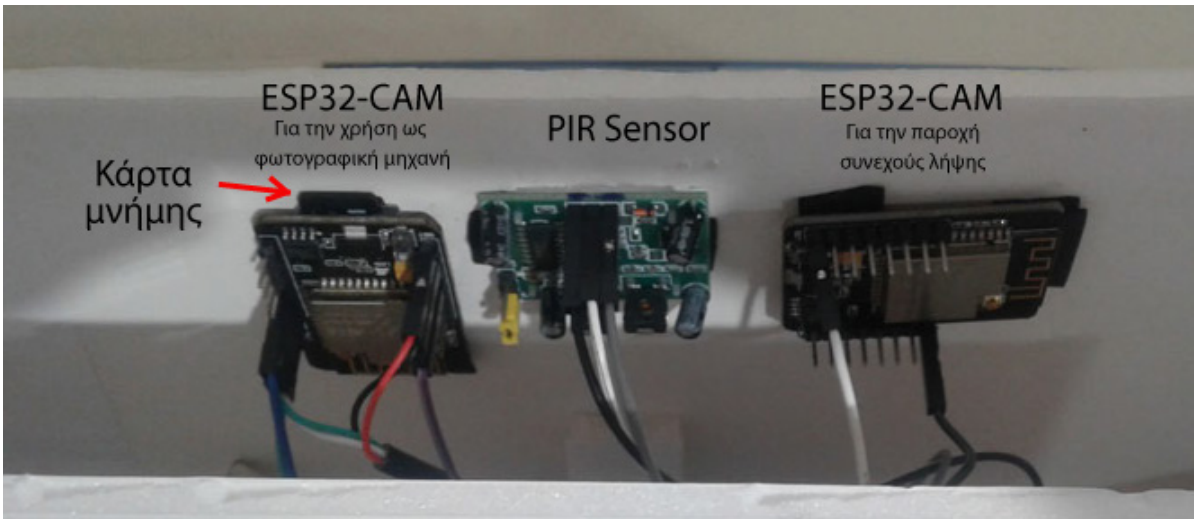
Στα παραρτήματα παρατίθενται και τα προγράμματα οδήγησης του συστήματος.

Στο σημείο αυτό θα πρέπει να επισημάνουμε πως με χρήση της εφαρμογής IFTTT (IF THIS THEN THAT) μόλις ο αισθητήρας ενεργοποιείται, αποστέλλεται στο κινητό του χρήστη κατάλληλη ειδοποίηση.



Σχήμα 3. Κύκλωμα συστήματος ελέγχου χώρου


ESP32-CAM
Για την χρήση ως
φωτογραφική μηχανή
ESP32-CAM
Για την παροχή
συνεχούς λήψης
PIR Sensor
Flash


Κάρτα
μνήμης
ESP32-CAM
Για την χρήση ως
φωτογραφική μηχανή
PIR Sensor
ESP32-CAM
Για την παροχή
συνεχούς λήψης

4.3.2 Οικονομική Ανάλυση

Ακολουθεί ο πίνακας που δείχνει το κόστος αγοράς των υλικών που απαιτήθηκαν για την κατασκευή του κυκλώματος ελέγχου του χώρου.

Πίνακας οικονομικής ανάλυσης 3: Οικονομική ανάλυση του κυκλώματος ελέγχου χώρου

Α/Α	ΠΕΡΙΓΡΑΦΗ	ΠΟΣΟΤΗΤΑ	ΚΟΣΤΟΣ
1	Espessif ESP32-CAM	2	38,0 €
2	PIR Sensor	1	4,4 €
3	Transistor 2N 3904	1	0,3 €
4	Resistor 1 kΩ	1	0,1 €
5	Resistor 10 kΩ	1	0,1 €
6	PL2303TA USB TTL to RS232 Converter Serial Cable	1	4,5 €
7	FTDI FT232RL USB to TTL SeriaConverter Adapter Module 5V and 3V	1	3,7 €
8	USB 2.0 MALE to Micro USB Cable	1	1,5 €
9	Πηγή Τάσης 5V	2	12,0 €
		ΣΥΝΟΛΟ	64,6 €

4.4 ΚΥΚΛΩΜΑ ΕΛΕΓΧΟΥ ΚΛΙΜΑΤΙΣΜΟΥ

4.4.1 Περιγραφή

Το κύκλωμα Ελέγχου του Κλιματισμού αποτελείται από:

- 1 Μικροελεγκτή ESP32
- 1 Ρελέ 1-καναλιού
- 1 Ρελέ 1-καναλιού
- 1 Αισθητήρα DHT11
- 1 Αισθητήρα Τάσης
- 1 Ανεμιστήρα

- Πηγή Τάσης 5V
- Πηγή Τάσης 9V

Ο προγραμματισμός του έγινε με χρήση της MicroPython (27) και αμέσως μετά παρουσιάζονται:

- Η σχηματική αναπαράσταση του κυκλώματος που κατασκευάστηκε
- Εικόνες από την κατασκευή του συστήματος
- Εικόνα από το web interface της ιστοσελίδας ελέγχου του συστήματος
- Η Οικονομική Ανάλυση του υποσυστήματος

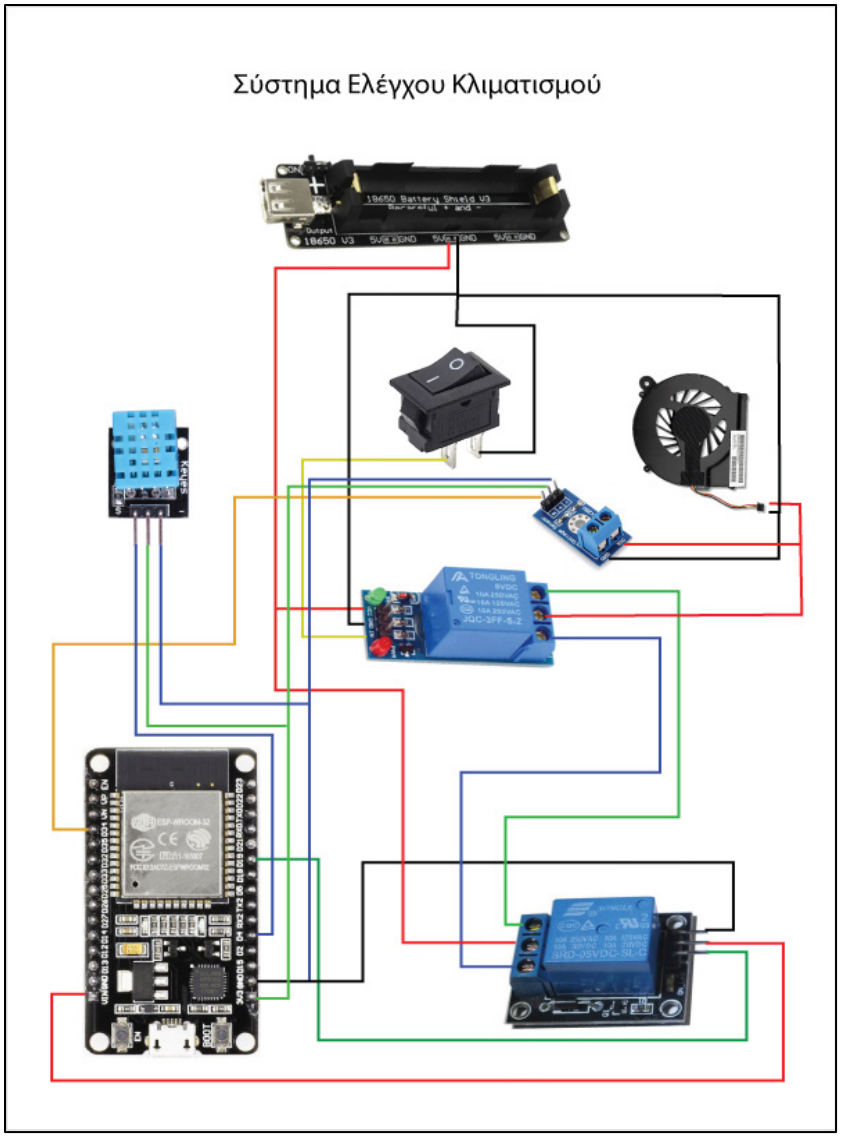
Στα παραρτήματα παρατίθενται και τα προγράμματα οδήγησης.

Η λειτουργία του συστήματος ελέγχεται μέσω του web interface στο οποίο παρέχεται και η πληροφορία για την κατάσταση στην οποία βρίσκεται ο κλιματισμός δηλαδή εάν βρίσκεται σε λειτουργία ή όχι. Η πληροφορία αυτή παρέχεται από την τιμή που λαμβάνει ο μικροελεγκτής από τον αισθητήρα τάσης. Με βάση την ένδειξη έχουμε την επιλογή να ανοίξουμε ή να κλείσουμε τον κλιματισμό πατώντας τα κουμπιά 

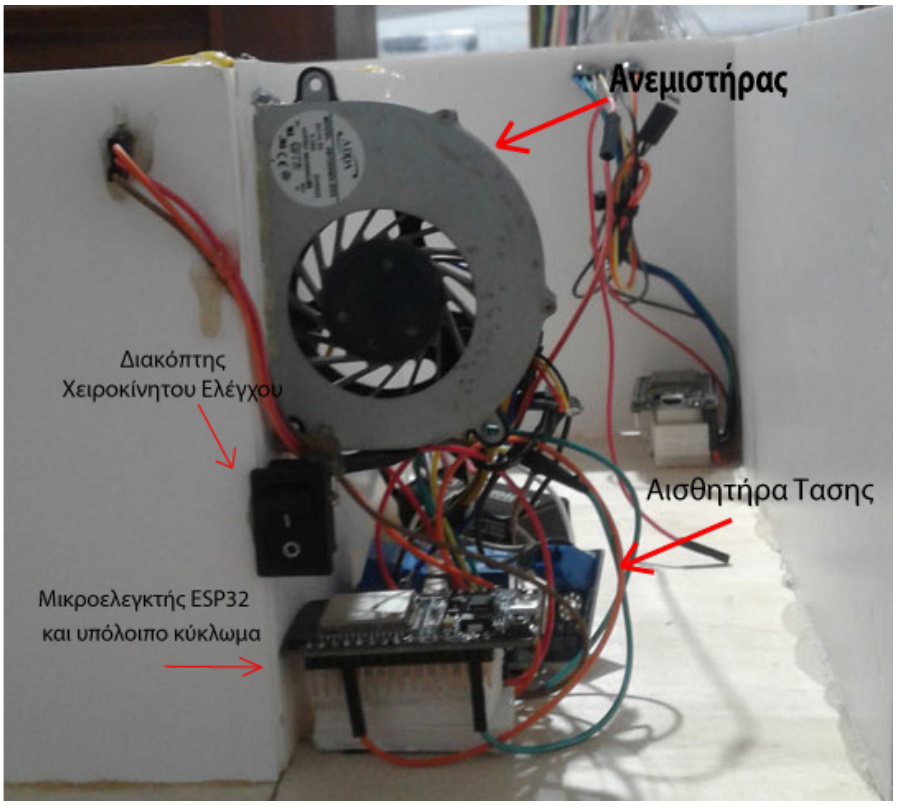
ή 

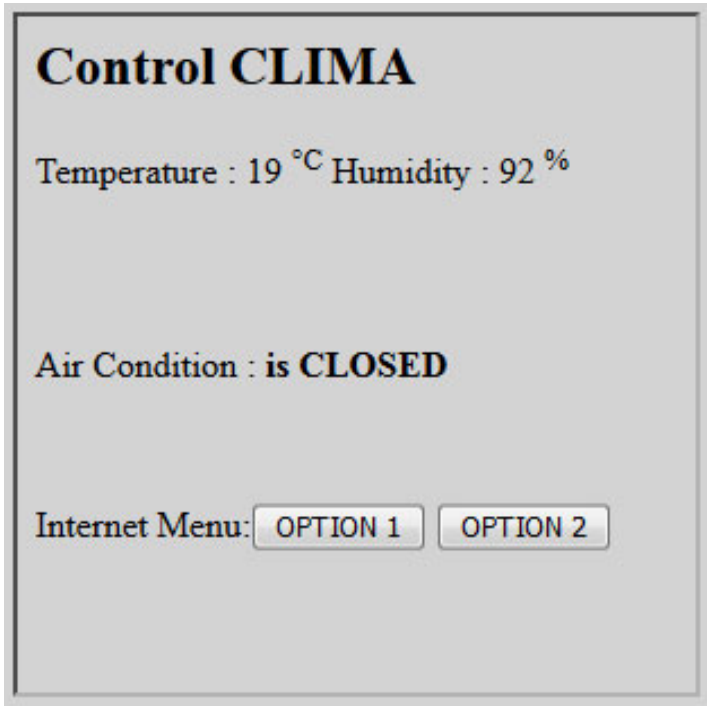
ανάλογα πάντα με την ένδειξη της κατάστασής του. Την κατάσταση του φωτισμού επίσης έχουμε τη δυνατότητα να την ελέγξουμε και χειροκίνητα μέσω του διακόπτη.

Αξίζει να σημειωθεί πως το υποσύστημα μπορεί να χρησιμοποιηθεί και για τον έλεγχο κατάστασης – λειτουργίας και άλλων ηλεκτρικών συσκευών όπως θερμοσίφωνα, ηλεκτρική κουζίνα κ.ά.



Σχήμα 4. Κύκλωμα συστήματος ελέγχου κλιματισμού


Ανεμιστήρας
Διακόπτης
Χειροκίνητου Ελέγχου
Αισθητήρα Τασης
Μικροελεγκτής ESP32
και υπόλοιπο κύκλωμα


Control CLIMA
Temperature : 19 °C Humidity : 92 %
Air Condition : is CLOSED
Internet Menu: OPTION 1 OPTION 2

4.4.2 Οικονομική Ανάλυση

Ακολουθεί ο πίνακας που δείχνει το κόστος αγοράς των υλικών που απαιτήθηκαν για την κατασκευή του κυκλώματος ελέγχου του κλιματισμού.

Πίνακας οικονομικής ανάλυσης 4: Οικονομική ανάλυση του κυκλώματος ελέγχου κλιματισμού

Α/Α	ΠΕΡΙΓΡΑΦΗ	ΠΟΣΟΤΗΤΑ	ΚΟΣΤΟΣ
1	Espessif ESP 32	1	13,5 €
2	Relay 1-Channel 5V	1	2,5 €
3	Relay 1-Channel 5V	1	2,0 €
4	Voltage Sensor	1	1,5 €
5	DHT11 Digital Temperature and Humidity Sensor	1	2,0 €
6	Cooler Fan for laptop	1	16,0 €
7	Rocker Switch 2pin On-Off	1	0,5 €
8	Πηγή Τάσης 5V	1	6,0 €
9	Πηγή Τάσης 9V	1	3,0 €
		ΣΥΝΟΛΟ	47,0 €

4.5 ΚΥΚΛΩΜΑ ΕΝΔΕΙΞΕΩΝ ΤΙΜΩΝ ΠΕΡΙΒΑΛΛΟΝΤΙΚΩΝ ΜΕΤΑΒΛΗΤΩΝ

4.5.1 Περιγραφή

Το κύκλωμα Ελέγχου των Τιμών των Περιβαλλοντικών Μεταβλητών αποτελείται από:

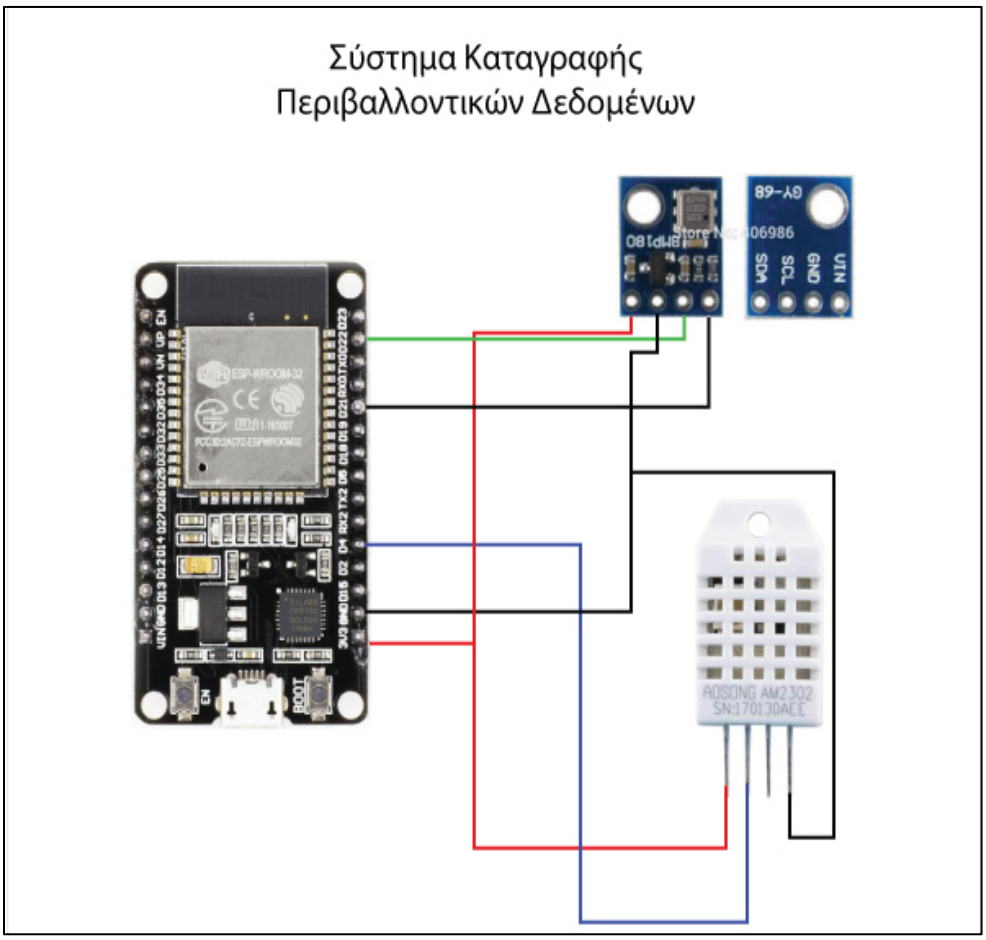
- 1 Μικροελεγκτή ESP32
- 1 Αισθητήρα DHT22
- 1 Αισθητήρα BMP180
- Πηγή Τάσης 5V

Ο προγραμματισμός του έγινε με χρήση της MicroPython (27) και αμέσως μετά παρουσιάζονται:

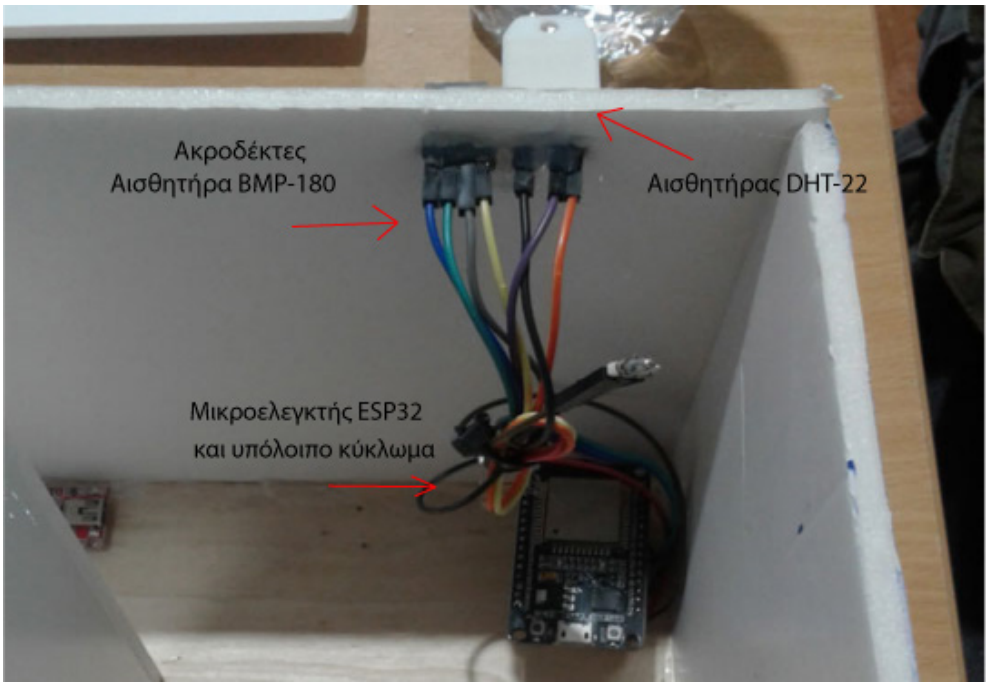
- Η σχηματική αναπαράσταση του κυκλώματος που κατασκευάστηκε
- Εικόνες από την κατασκευή του συστήματος
- Εικόνα από το web interface της ιστοσελίδας ελέγχου του συστήματος
- Η Οικονομική Ανάλυση του υποσυστήματος

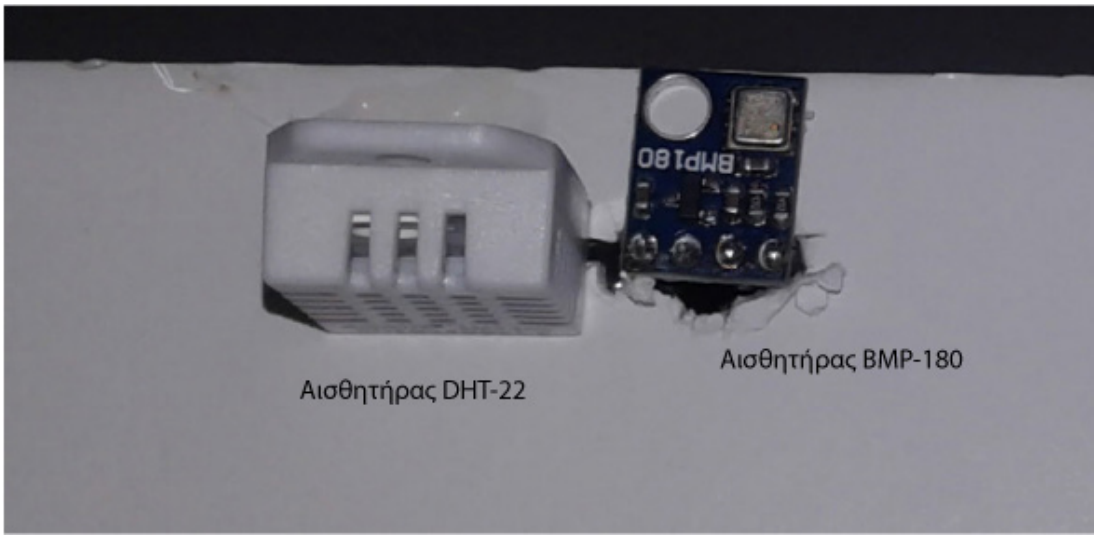
Στα παραρτήματα παρατίθενται και τα προγράμματα οδήγησης της συσκευής.

Κατά τον προγραμματισμό του υποσυστήματος απαιτήθηκε το αρχείο-βιβλιοθήκη **bmp180.py** το οποίο υπάρχει στο διαδίκτυο και σε διάφορες εκδόσεις από τις οποίες στην εργασία μας χρησιμοποιήσαμε την έκδοση πού δίνεται στην ιστοσελίδα με URL: http://www.learnmicropython.com/esp32/a-bmp180-micropython-example-on-an-esp32.php.



Σχήμα 5. Κύκλωμα συστήματος ελέγχου περιβαλλοντικών δεδομένων


Ακροδέκτες
Αισθητήρα BMP-180
Αισθητήρας DHT-22
Μικροελεγκτής ESP32
και υπόλοιπο κύκλωμα


Αισθητήρας DHT-22
Αισθητήρας BMP-180


Περιβαλλοντικές ενδείξεις
Temperature 19.1 °C Humidity 56.2 % Pressure 101562.2 Pa

4.5.2 Οικονομική ανάλυση

Ακολουθεί ο πίνακας που δείχνει το κόστος αγοράς των υλικών που απαιτήθηκαν για την κατασκευή του κυκλώματος ελέγχου των τιμών των περιβαλλοντικών μεταβλητών θερμοκρασία, ατμοσφαιρική πίεση και υγρασία ατμόσφαιρας.

Πίνακας οικονομικής ανάλυσης 5. Οικονομική ανάλυση του κυκλώματος Ελέγχου των Τιμών των Περιβαλλοντικών Μεταβλητών

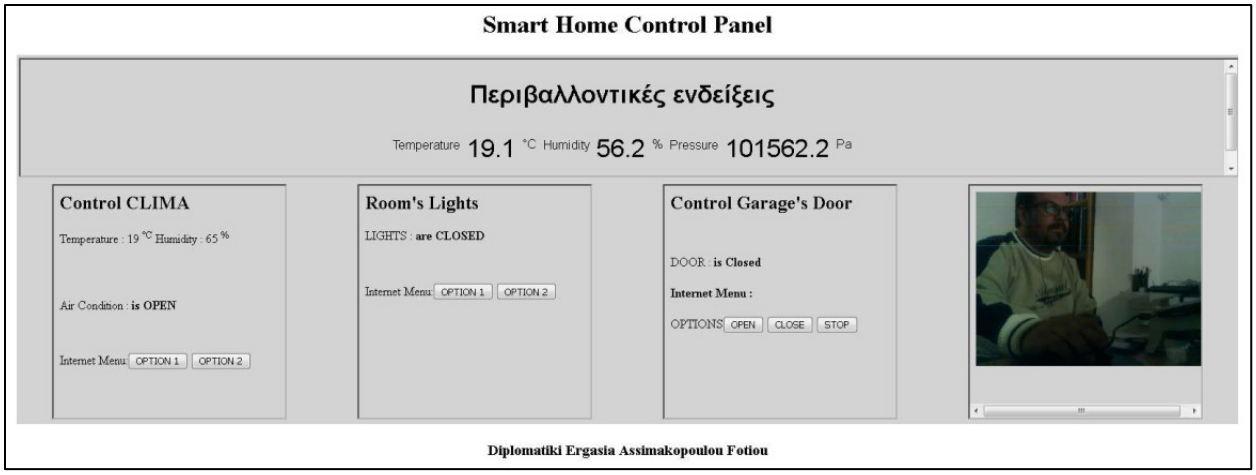
Α/Α	ΠΕΡΙΓΡΑΦΗ	ΠΟΣΟΤΗΤΑ	ΚΟΣΤΟΣ
1	Espessif ESP 32	1	13,5 €
2	DHT22/AM2302 Digital Temperature and Humidity Sensor	1	6,8 €
3	BMP180 Digital Barometric Pressure Sensor	1	7,0 €
4	Πηγή Τάσης 5V	1	6,0 €
		ΣΥΝΟΛΟ	33,3 €

4.6 ΙΣΤΟΣΕΛΙΔΑ ΕΛΕΓΧΟΥ ΤΩΝ ΥΠΟΣΥΣΤΗΜΑΤΩΝ

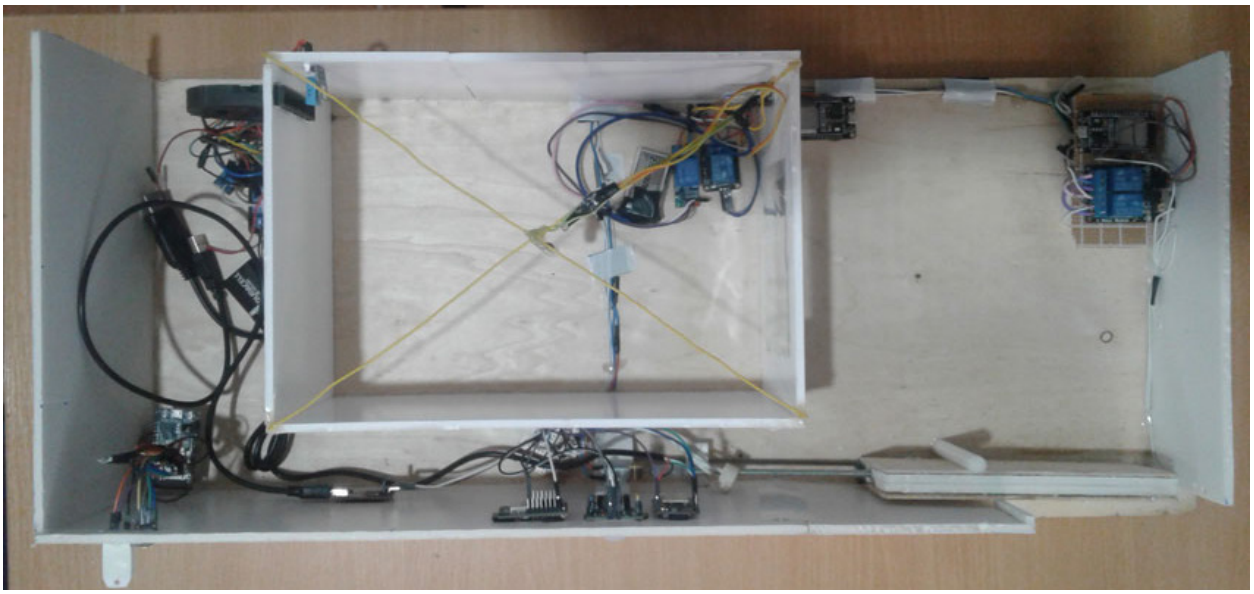
Με την κατασκευή της κατάλληλης ιστοσελίδας μπορούμε να ελέγχουμε και μέσω διαδικτύου τα υποσυστήματα της εφαρμογής.

Η κατασκευή της έγινε με χρήση του Notepad++, με την εφαρμογή απλού τμήματος CSS, ενώ για την ενσωμάτωση των ιστοσελίδων του κάθε επί μέρους υποσυστήματος σε μία ενιαία σελίδα χρησιμοποιήθηκε η ετικέτα (tag) **<iframe …>…</ iframe >** όπως φαίνεται και στον αντίστοιχο κώδικα στο παράρτημα. Πρακτικά, κάθε *iframe* ενσωματώνει τον κώδικα της ιστοσελίδας ενός υποσυστήματος.

Ακολουθεί ένα στιγμιότυπο αυτής καθώς και εικόνα από τελικό στάδιο της πλήρης ανάπτυξης της μακέτας.



Σχήμα 6. Στιγμιότυπο ιστοσελίδας πίνακα ελέγχου



5. ΣΥΜΠΕΡΆΣΜΑΤΑ – ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

Στην παρούσα εργασία μπορέσαμε να αναδείξουμε πώς είναι δυνατόν να αναπτυχθούν μηχανισμοί αυτοματισμού με χαμηλό κόστος και υψηλή αξιοπιστία, οι οποίοι να διευκολύνουν τη διαχείριση καθημερινών λειτουργιών, τόσο για την παρακολούθηση της κατάστασης ενός έξυπνου σπιτιού όσο και για την πραγματοποίηση ενεργειών (π.χ. έλεγχος γκαραζόπορτας). Επιπλέον αναδείξαμε πως στα ίδια συστήματα μπορεί να συνδυαστεί ο έλεγχος τους ταυτόχρονα και μέσω διαδικτύου και με την φυσική μας αλληλεπίδραση με το σύστημα.

Η δυνατότητα που μας παρέχει η δημιουργία ενός ολοκληρωμένου συστήματος τέτοιου τύπου θα μας βοηθήσει και στην εξοικονόμηση ενέργειας άρα και χαμηλότερου κόστος διαβίωσης, αλλά και με μεγαλύτερη ασφάλεια.

Εκτός από τη δημιουργία και κατασκευή των υποσυστημάτων που αναπτύξαμε παραπάνω, η ολοκλήρωση της αυτοματοποίησης ενός «Έξυπνου Σπιτιού» θα μπορούσε να περιλάβει τη δημιουργία και άλλων υποσυστημάτων όπως ενδεικτικά:

- Σύστημα αυτόματου ποτίσματος εξωτερικών εκτάσεων.
- Σύστημα ελέγχου για τις τέντες ανάλογα με τις καιρικές συνθήκες.
- Σύστημα ελέγχου διαρροής στο δίκτυο ύδρευσης.

με βασικούς παράγοντες πάντα

- Την οικονομία,
- Την ευκολία χρήσης,
- Την αξιοπιστία και
- Την ασφάλεια.

6. ΒΙΒΛΙΟΓΡΑΦΙΑ

1. **Πουλάκης, Εμμανουήλ.** *Προγραμματίζοντας με τον μικροελεγκτή Arduino.* Ηράκλειο : s.n., 2015.

2. **Introduction to MicroPython. [Ηλεκτρονικό] https://docs.pycom.io/gettingstarted/programming/micropython/.**

3. MicroPython documentation. [Ηλεκτρονικό] https://docs.micropython.org/en/latest/index.html.

4. Tutorials, Random Nerd. MicroPython Programming Basics with ESP32 and ESP8266. [Ηλεκτρονικό] https://randomnerdtutorials.com/micropython-programming-basics-esp32-esp8266/.

5. Αγγελίδης, Μάριος. HTML 5. [Ηλεκτρονικό] https://zenos.gr/html5/.

6. Τσελίκας, Νικόλαος Δ. Ανάπτυξη Εφαρμογών Κινητών Τερματικών.

7. Tutorials, Random Nerd. ESP32 Pinout Reference: Which GPIO pins should you use? [Ηλεκτρονικό] https://randomnerdtutorials.com/esp32-pinout-reference-gpios/.

8. [Ηλεκτρονικό] https://en.wikipedia.org/wiki/ESP32.

9. Espressif. ESP32 Soc Overview . [Ηλεκτρονικό] https://www.espressif.com/en/products/hardware/esp32/overview.

10. ESP32 ESP-32S Development Board WiFi+Bluetooth. [Ηλεκτρονικό] https://www.cableworks.gr/ilektronika/arduino-and-microcontrollers/microcontrollers/esp32/esp32-esp-32s-development-board-wifibluetooth-dual-core-ultralow-power-consumption-microcontoller/.

11. ESP32-CAM - ESP32 WIFI Bluetooth Development Board With OV2640 Camera Module. [Ηλεκτρονικό] https://www.cableworks.gr/ilektronika/arduino-and-microcontrollers/microcontrollers/esp32-wifi-bluetooth-development-board-with-ov2640-camera-module/.

12. Camera Module Based on ESP32. [Ηλεκτρονικό] https://grobotronics.com/camera-module-based-on-esp32.html.

13. GL5516. [Ηλεκτρονικό] https://www.cableworks.gr/ilektronika/arduino-and-microcontrollers/components-and-ic/gl5516-5mm-photoresistor-ldr-light-dependent-resistor-light-sensor-for-arduino/.

14. PIR Motion Detector Module HC-SR501. [Ηλεκτρονικό] https://www.cableworks.gr/ilektronika/arduino-and-microcontrollers/mcu-and-components/distance/pir-motion-detector-module-hc-sr501-for-arduino/.

15. BMP180 Digital Barometric Pressure Sensor. [Ηλεκτρονικό] https://www.cableworks.gr/ilektronika/arduino-and-microcontrollers/mcu-and-components/pressure/bmp180-digital-barometric-pressure-sensor-for-arduino-gy-68/.

16. Bosch. BMP180 Digital Pressure Sensor. [Ηλεκτρονικό] https://cdn-shop.adafruit.com/datasheets/BST-BMP180-DS000-09.pdf.

17. DHT11 Digital Temperature and Humidity Sensor. [Ηλεκτρονικό] https://www.cableworks.gr/ilektronika/arduino-and-microcontrollers/mcu-and-components/temperature/dht11-digital-temperature-and-humidity-sensor-for-arduino/.

18. DHT22/AM2302 Digital Temperature and Humidity Sensor. [Ηλεκτρονικό] https://www.cableworks.gr/ilektronika/arduino-and-microcontrollers/mcu-and-components/temperature/dht22-am2302-digital-temperature-and-humidity-sensor-for-arduino/.

19. Voltage Detection Module - Voltage Sensor. [Ηλεκτρονικό] https://www.cableworks.gr/ilektronika/arduino-and-microcontrollers/mcu-and-components/current-voltage/voltage-detection-module-voltage-sensor-for-arduino/.

20. 5V 1-Channel Relay Module Board. [Ηλεκτρονικό] https://www.cableworks.gr/ilektronika/arduino-and-microcontrollers/relays/5v-relays/5v-1-channel-relay-module-board-shield-for-arduino/.

21. 5V 1-Channel Small Relay Module Board. [Ηλεκτρονικό] https://www.cableworks.gr/ilektronika/arduino-and-microcontrollers/relays/5v-relays/5v-1-channel-small-relay-module-board-for-arduino/.

22. 2-Channel Relay Module Board 5V. [Ηλεκτρονικό] https://www.cableworks.gr/ilektronika/arduino-and-microcontrollers/relays/5v-relays/5v-2-channel-relay-module-board-for-arduino/.

23. FTDI FT232RL USB to TTL Serial Converter Adapter Module 5V and 3.3V. [Ηλεκτρονικό] https://www.cableworks.gr/ilektronika/arduino-and-microcontrollers/modules/converter/ftdi-ft232rl-usb-to-ttl-serial-converter-adapter-module-5v-and-3.3v-for-arduino/.

24. PL2303TA USB TTL to RS232 Converter Serial Cable. [Ηλεκτρονικό] https://www.cableworks.gr/ilektronika/arduino-and-microcontrollers/modules/converter/pl2303ta-usb-ttl-to-rs232-converter-serial-cable/.

25. 6V DC Metal Mini Motor GA12-N20 30RPM Speed. [Ηλεκτρονικό] https://www.cableworks.gr/ilektronika/arduino-and-microcontrollers/motors/dc/6v-dc-metal-mini-motor-ga12-n20-30rpm-speed/.

26. Philips. 2N3904 Datasheet. [Ηλεκτρονικό] https://pdf1.alldatasheet.com/datasheet-pdf/view/15077/PHILIPS/2N3904.html.

27. MicroPython. Quick reference for the ESP32. [Ηλεκτρονικό] http://docs.micropython.org/en/latest/esp32/quickref.html.

28. Sample, Arduino. CameraWebServer. [Ηλεκτρονικό] http://acoptex.com/project/10297/basics-project-084a-esp32-cam-development-board-with-camera-camera-web-server-with-arduino-ide-at-acoptexcom/#sthash.w56WQY6w.dpbs.

29. Tutorials, Random Nerd. ESP32-CAM PIR Motion Detector with Photo Capture. [Ηλεκτρονικό] https://randomnerdtutorials.com/esp32-cam-pir-motion-detector-photo-capture/.

30. Limit Switch SS-5GL. [Ηλεκτρονικό] https://www.cableworks.gr/3d-printers/build-it-yourself/end-stops/limit-switch-ss-5gl-lever-arm-normally-open-close/.

7. ΠΑΡΑΡΤΗΜΑΤΑ

7.1 ΕΓΚΑΤΑΣΑΣΗ ΛΟΓΙΣΜΙΚΟΥ

7.1.1 Arduino IDE

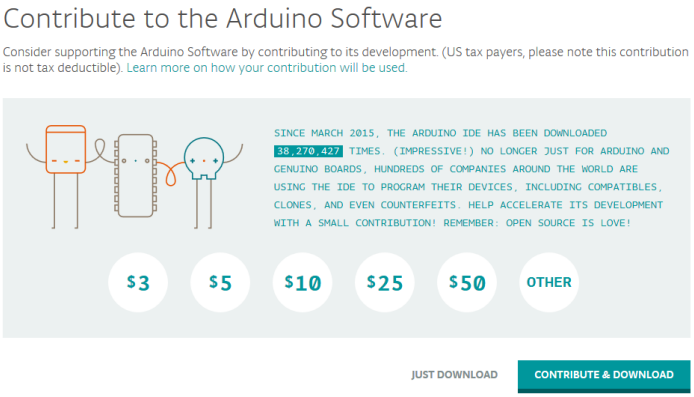
Για να εγκαταστήσουμε το Arduino IDE σε περιβάλλον Windows ακολουθούμε τις εξής οδηγίες:

- Χρησιμοποιώντας έναν browser όπως π.χ. ο Chrome επισκεπτόμαστε την επίσημη σελίδα του Arduino (https://www.arduino.cc/).

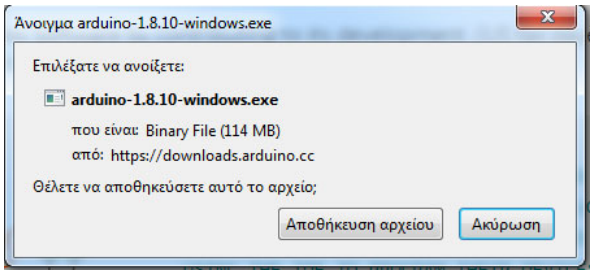
- Στη συνέχεια, επιλέγουμε την κατηγορία **SOFTWARE/DOWNLOADS**.

Download the Arduino IDE

- Από το παράθυρο που εμφανίζεται στα δεξιά της οθόνης επιλέγουμε το **Windows Installer** και στη συνέχεια την επιλογή **Just Download**.

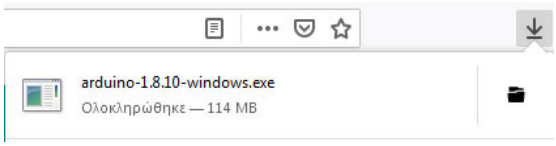


- Στο παράθυρο που ανοίγει πατάμε **Αποθήκευση αρχείου**



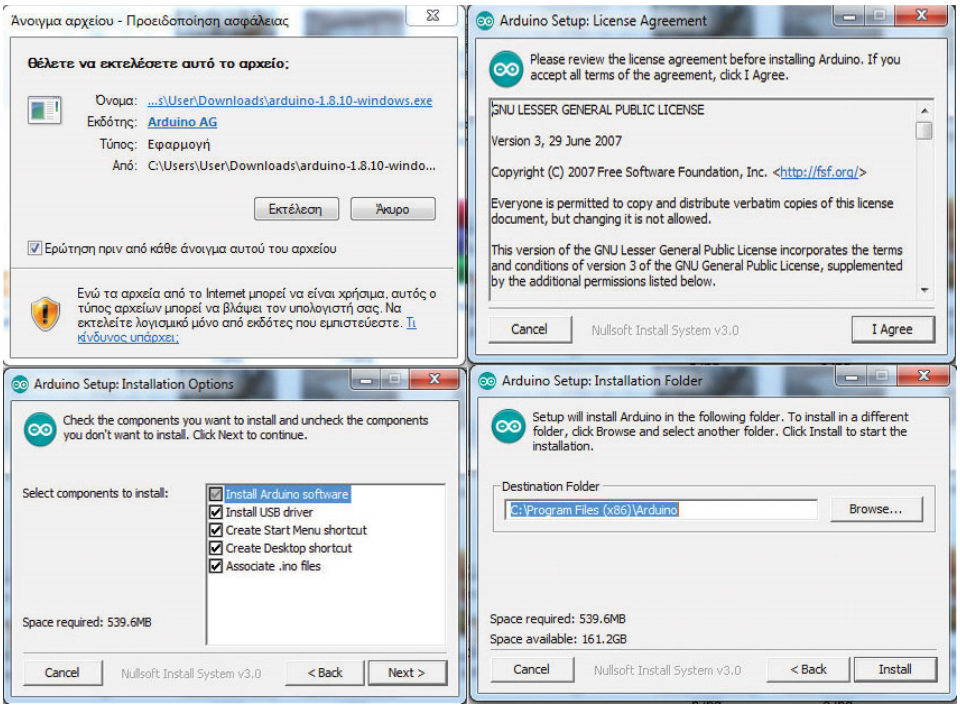
- Στο επόμενο παράθυρο επιλέγουμε τον φάκελο στον οποίο θέλουμε να γίνει η αποθήκευση και πατάμε **Αποθήκευση**

- Το αρχείο θα αρχίσει να κατεβαίνει.



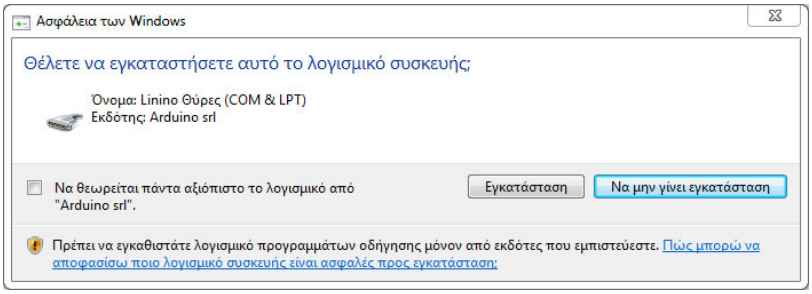
- Μετά την ολοκλήρωση της λήψης κάνουμε διπλό κλικ στο αρχείο το οποίο το βρίσκουμε στον φάκελο που το αποθηκεύσαμε σε προηγούμενο βήμα

- Στα ακόλουθα παράθυρα διαλόγου που εμφανίζονται

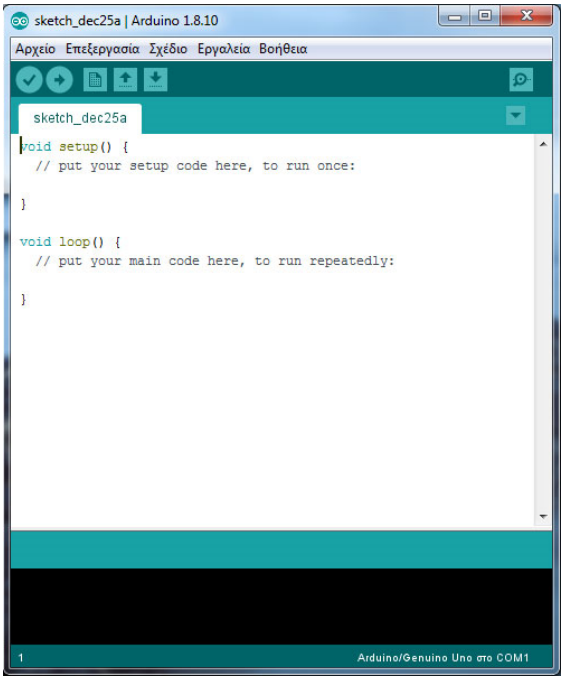


επιλέγουμε **Εκτέλεση, I Agree**, **Next**, και **Install** αντίστοιχα.

- Σε όλα τα επόμενα παράθυρα του ακόλουθου τύπου πατάμε **Εγκατάσταση**



- Μόλις η εγκατάσταση έχει ολοκληρωθεί με επιτυχία και το εικονίδιο της συντόμευσης για το Arduino IDE έχει εμφανιστεί στην επιφάνεια εργασίας. Κάνοντας διπλό κλικ στο εικονίδιο θα εμφανιστεί το ακόλουθο παράθυρο στο οποίο πλέον μπορούμε να δημιουργούμε εφαρμογές.



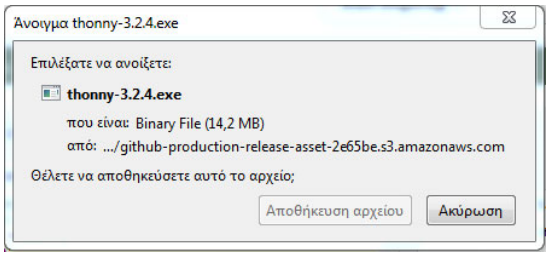
7.1.2 THONNY IDE

Όταν θέλουμε να προγραμματίσουμε έναν μικροελεγκτή ESP32 με χρήση της MicroPython πρέπει να χρησιμοποιήσουμε κατάλληλο περιβάλλον. Έτσι, όπως έχει αναφερθεί, ένα από τα περιβάλλοντα που χρησιμοποιήθηκαν στην παρούσα εργασία είναι το Thonny IDE.

Για να εγκαταστήσουμε το Thonny IDE σε περιβάλλον Windows ακολουθούμε τις εξής οδηγίες:

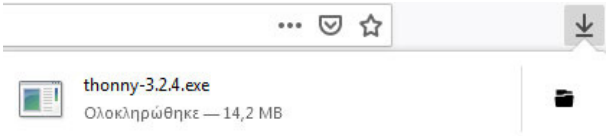
- Πηγαίνουμε στη διεύθυνση https://thonny.org
- Επιλέγουμε την έκδοση για Windows και περιμένουμε μικρό χρονικό διάστημα μέχρι να κατεβεί.

- Στο παράθυρο που ανοίγει πατάμε **Αποθήκευση αρχείου**

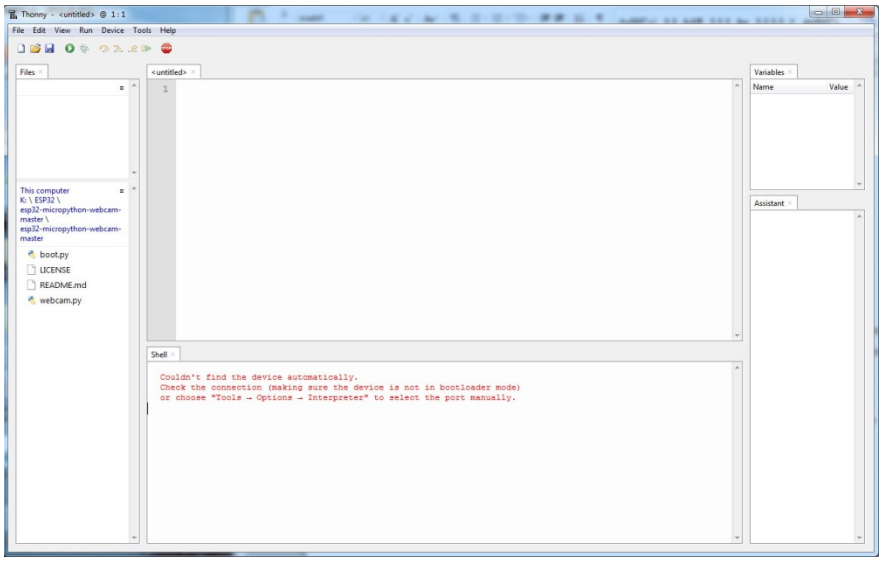


- Στο επόμενο παράθυρο επιλέγουμε τον φάκελο στον οποίο θέλουμε να γίνει η αποθήκευση και πατάμε **Αποθήκευση**

- Το αρχείο θα αρχίσει να κατεβαίνει.



- Μετά την ολοκλήρωση της λήψης κάνουμε διπλό κλικ στο αρχείο το οποίο το βρίσκουμε στον φάκελο που το αποθηκεύσαμε σε προηγούμενο βήμα και ακολουθώντας τις οδηγίες ολοκληρώνουμε την εγκατάσταση.
- Μόλις η εγκατάσταση έχει ολοκληρωθεί με επιτυχία και το εικονίδιο της συντόμευσης για το Thonny IDE έχει εμφανιστεί στην επιφάνεια εργασίας. Κάνοντας διπλό κλικ στο εικονίδιο θα εμφανιστεί το ακόλουθο παράθυρο στο οποίο πλέον μπορούμε να δημιουργούμε εφαρμογές.

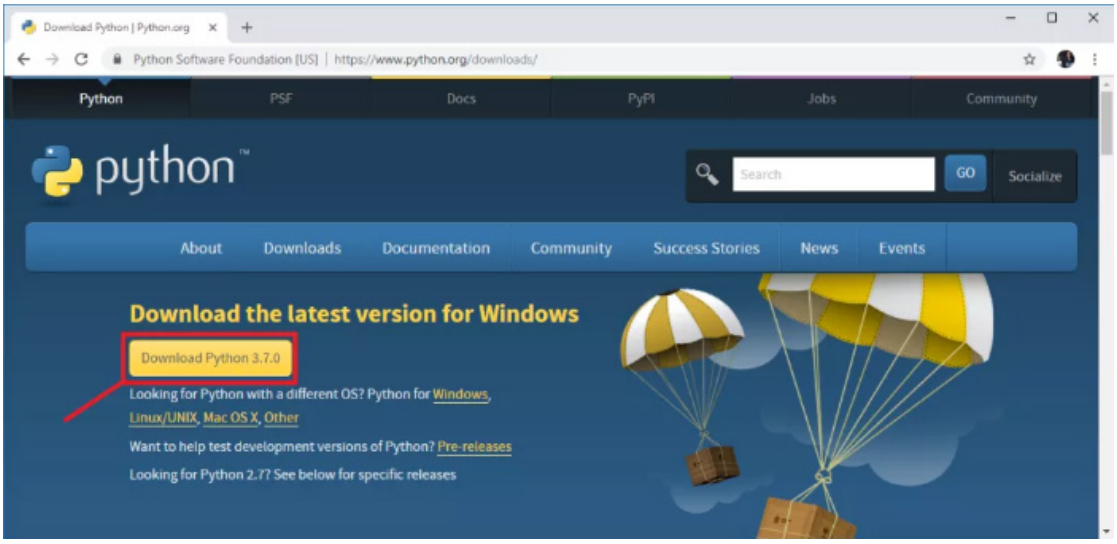


7.1.3 uPyCraft IDE

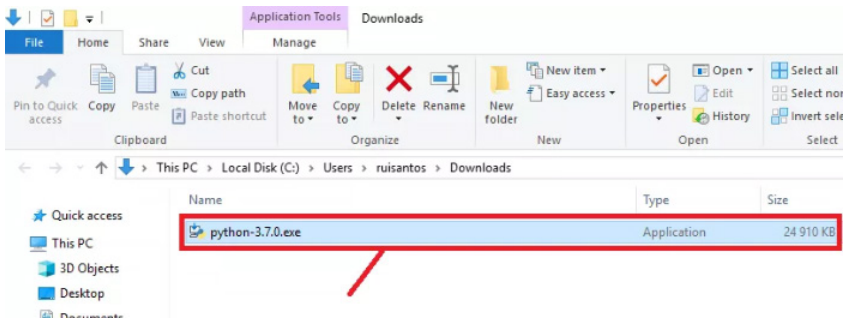
Το δεύτερο περιβάλλον που χρησιμοποιήθηκε για τον προγραμματισμό των μικροελεγκτών ESP32 με χρήση της MicroPython όπως προείπαμε είναι το uPyCraft IDE.

Πριν όμως προχωρήσουμε στην εγκατάσταση του θα πρέπει να έχουμε εγκαταστήσει στον υπολογιστή μας την τελευταία έκδοση της Python 3.7.X. Εάν δεν το έχουμε πράξει τότε μπορούμε να ακολουθήσουμε τα ακόλουθα βήματα.

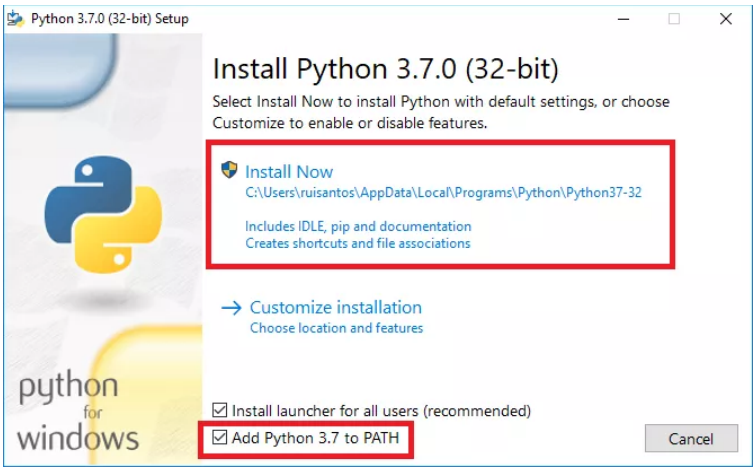
- Πηγαίνουμε στην κατάλληλη σελίδα www.python.org/downloads και κατεβάζουμε το αρχείο εγκατάστασης.



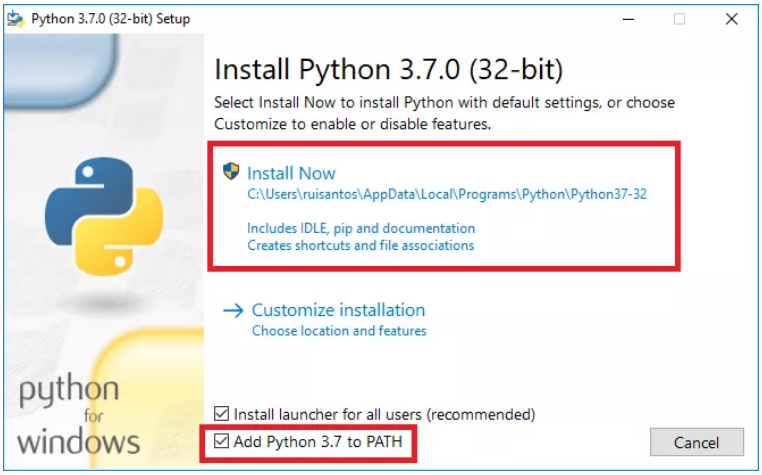
- Μόλις ολοκληρωθεί το κατέβασμα του αρχείου εγκατάστασης στον υπολογιστή μας, θα έχουμε στο φάκελο που ορίσαμε ένα αρχείο με όνομα της μορφής **python-3.7.X.exe**. Πατάμε διπλό κλικ και ανοίγουμε το αρχείο αυτό.

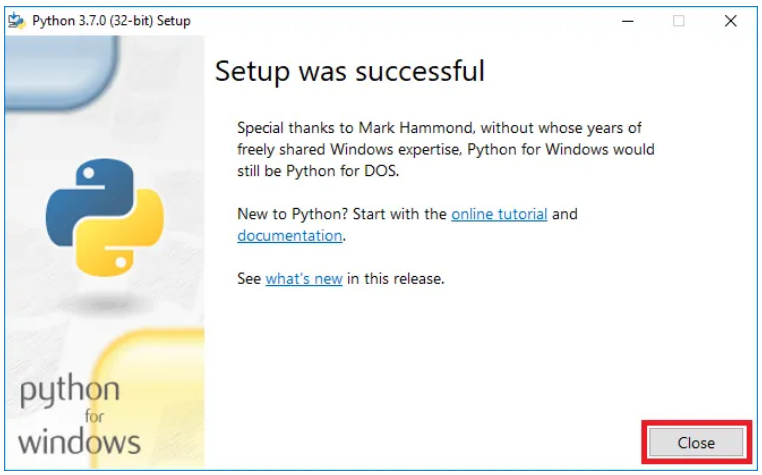


- Ενεργοποιούμε την επιλογή "**Add Python 3.7 to PATH**" στο κάτω μέρος της οθόνης που έχει ανοίξει και μετά πατάμε "**Install Now**".



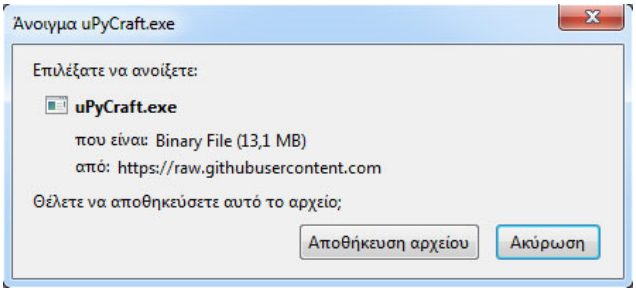
- Μετά από μερικά δευτερόλεπτα αναμονής ολοκληρώνεται η διαδικασία εγκατάστασης και πατώντας το πλήκτρο **Close** στο παράθυρο με το μήνυμα "**Setup was successful**" κλείνουμε τη διαδικασία εγκατάστασης.



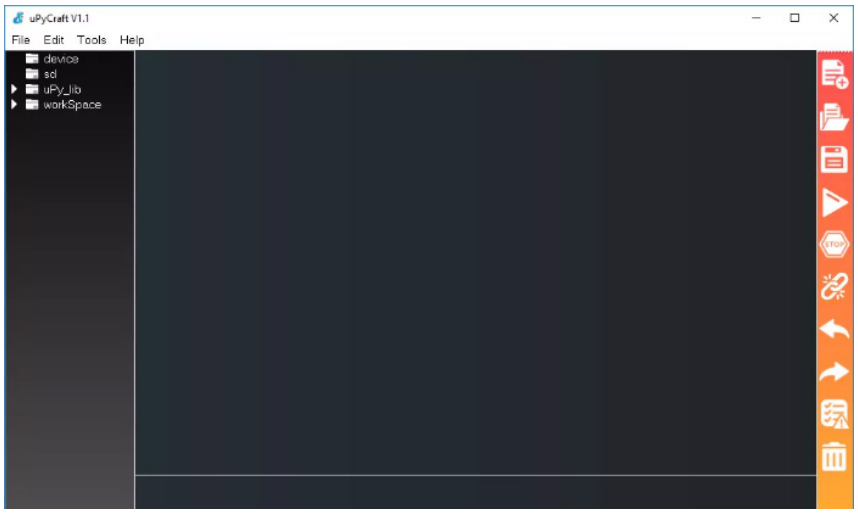


Τώρα μπορούμε να περάσουμε στη διαδικασία εγκατάστασης του uPyCraft IDE ακολουθώντας τα παρακάτω βήματα.

- Πηγαίνουμε στο link https://randomnerdtutorials.com/uPyCraftWindows.
- Στο παράθυρο που ανοίγει πατάμε **Αποθήκευση Αρχείου**



- Στο επόμενο παράθυρο επιλέγουμε τον φάκελο στον οποίο θέλουμε να γίνει η αποθήκευση και πατάμε **Αποθήκευση αρχείου**
- Το αρχείο θα αρχίσει να κατεβαίνει.
- Μόλις τελειώσει το κατέβασμα στον φάκελο προορισμού θα βρούμε ένα αρχείο με όνομα της μορφής uPyCraft_VX.exe.
- Ανοίγοντας το αρχείο αυτό βρισκόμαστε στο περιβάλλον του uPyCraft IDE με το οποίο μπορούμε να εγκαταστήσουμε το firmware της MicroPython στους μικροελεγκτές μας αλλά και να τους προγραμματίσουμε.



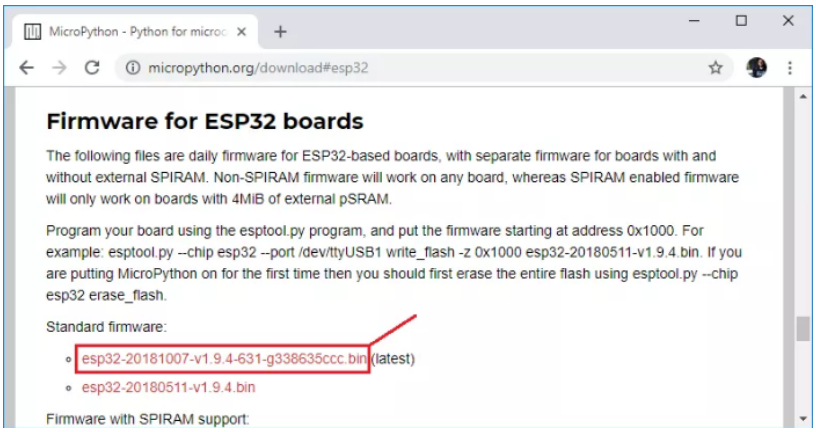
7.1.4 Εγκατάσταση του Micropython Firmware

Η εγκατάσταση του Micropython Firmware είναι απαραίτητη προϋπόθεση προκειμένου να έχει τη δυνατότητα ο μικροελεγκτής ESP32 να εκτελέσει προγράμματα MicroPython.
Για να το πραγματοποιηθεί η εγκατάσταση, μπορούμε να ακολουθήσουμε μία από τις δύο εναλλακτικές διαδρομές που περιγράφονται στις ακόλουθες υποενότητες.

7.1.4.1 Με χρήση του uPyCraft IDE

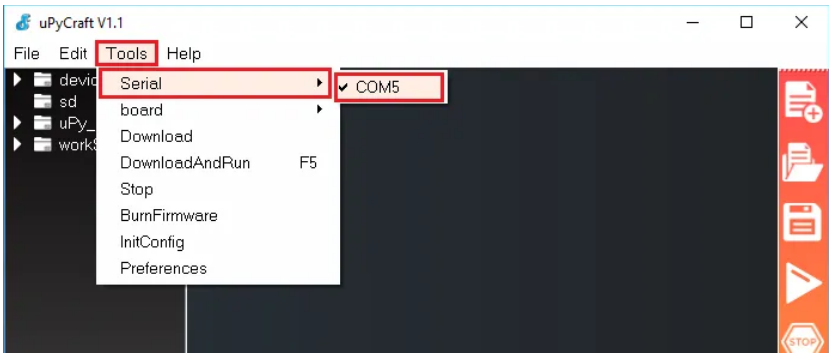
Για να μπορέσουμε να εγκαταστήσουμε το Micropython firmware μέσω του uPyCraft IDE, θα πρέπει κατ' αρχήν να έχουμε εγκαταστήσει το ίδιο το uPyCraft IDE. Μετέπειτα ακολουθούμε τα παρακάτω βήματα:

- Αρχικά κατεβάζουμε την τελευταία έκδοση του MicroPython firmware για το ESP32, πηγαίνοντας στη σελίδα M http://micropython.org/download#esp32 icroPython Downloads page[1] και στον τομέα που αφορά τις πλακέτες ESP32.



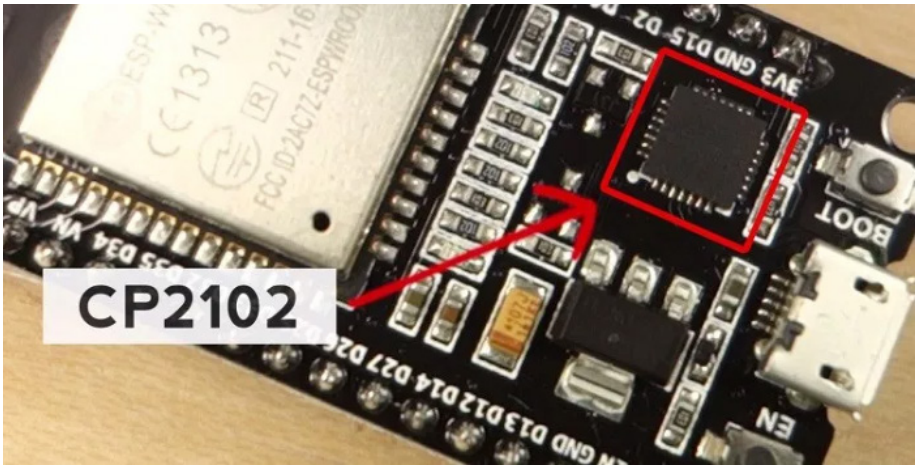
- Ανοίγουμε το uPyCraft και στο μενού **Tools>Serial** επιλέγουμε τη θύρα επικοινωνίας (COM port) στο οποίο είναι συνδεδεμένη η πλακέτα μας.

[1] http://micropython.org/download#esp32

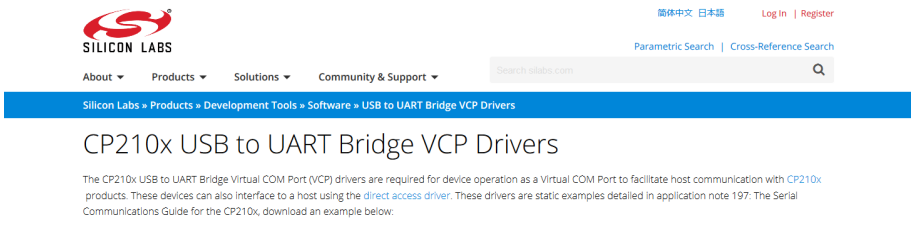


Σημαντικό: Όταν συνδέουμε την πλακέτα ESP32 στον υπολογιστή και δεν βρίσκουμε στο uPyCraft διαθέσιμη θύρα τότε:

- Είτε έχουμε χρησιμοποιήσει καλώδιο που δεν έχει δυνατότητα μεταφοράς δεδομένων ή
- Είτε πρέπει να εγκαταστήσουμε τον απαραίτητο driver για το chipset της πλακέτας. Αυτό γίνεται ως εξής:
  - ✓ βρίσκουμε το όνομα του chipset το οποίο βρίσκεται μπροστά από το κουμπί **BOOT** της πλακέτας

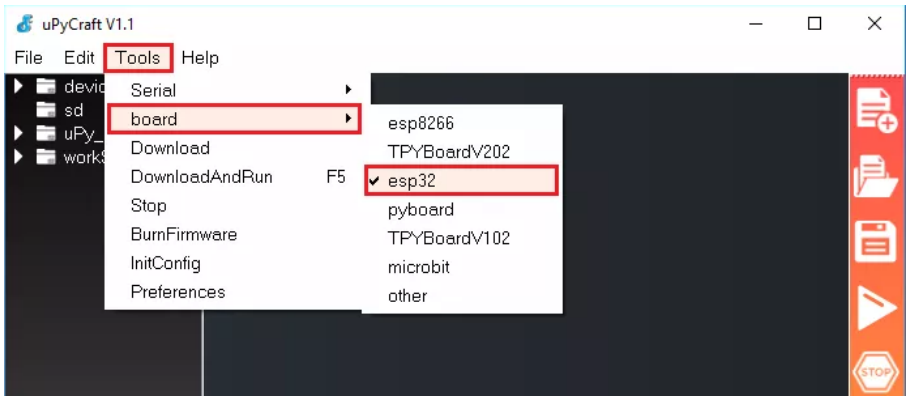


  - ✓ Εντοπίζουμε, κατεβάζουμε και εγκαθιστούμε τον σωστό driver. Αν π.χ. το chipset της πλακέτας που χρησιμοποιούμε είναι το CP2102 μπορούμε να πραγματοποιήσουμε τη λήψη μέσω της ιστοσελίδας της Silicon Labs



ενώ στη γενική περίπτωση, μπορούμε να εντοπίσουμε τους drivers μέσω κάποιας μηχανής αναζήτησης (π.χ. της Google)

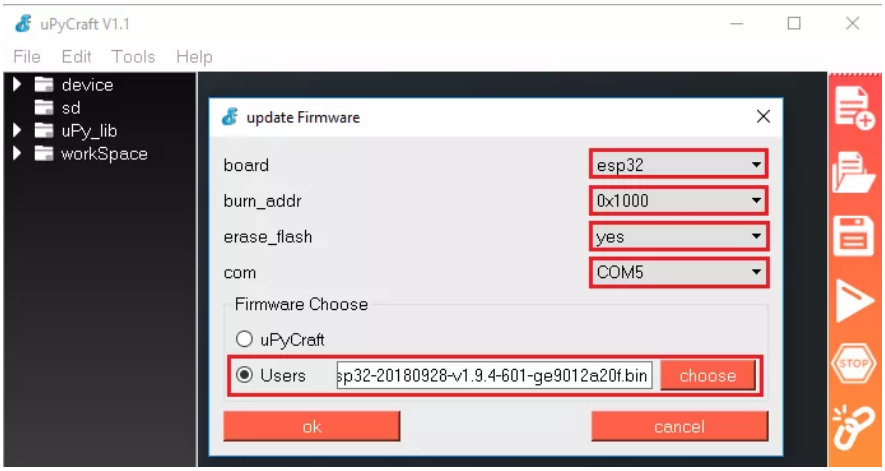
- Από το μενού **Tools>Board** επιλέγουμε το **esp32**



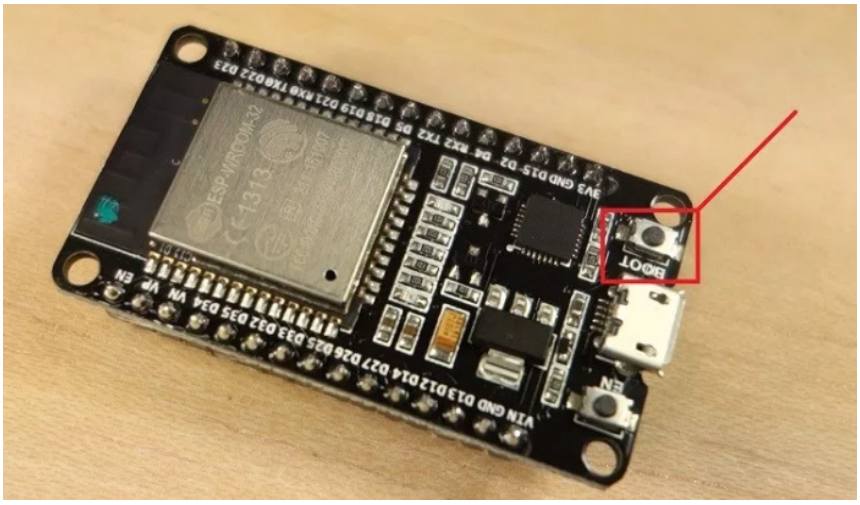
- Τέλος, στο μενού **Tools** επιλέγουμε **BurnFirmware**

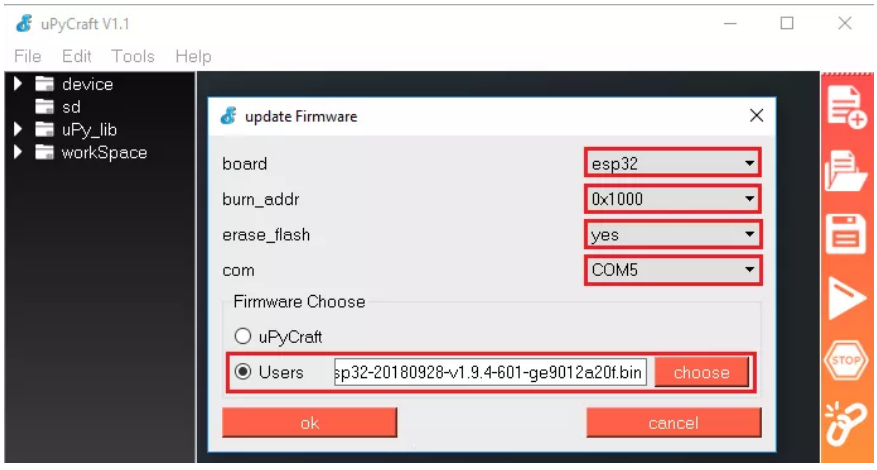
- Στη σελίδα που ανοίγει επιλέγουμε:
  - board: **esp32**
  - burn_addr: **0x1000**
  - erase_flash: **yes**
  - com: **COMx** (τον αριθμό της COM που έχουμε συνδέσει την πλακέτα μας)

- Firmware Choose: επιλέγουμε **Users** και πατώντας **choose** επιλέγουμε το firmware που κατεβάσαμε σε προηγούμενο βήμα



- Έχοντας πατημένο το κουμπί **BOOT** στο ESP32 επιλέγουμε το **ok**





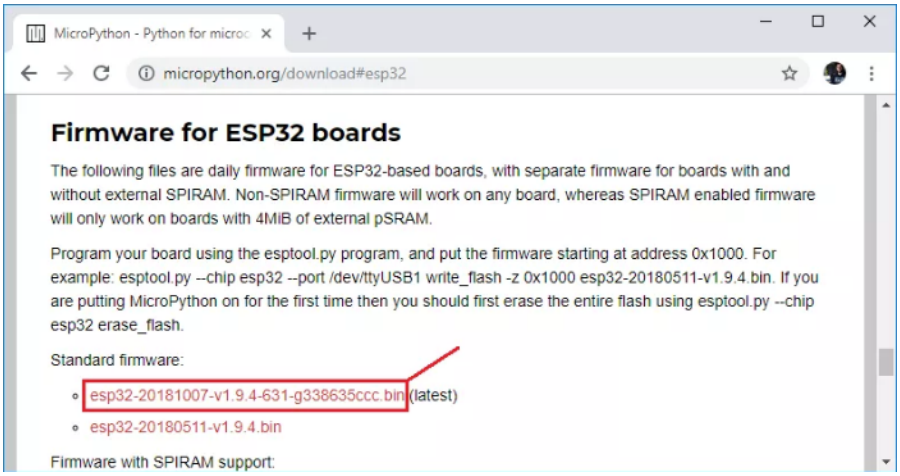
- Όταν ξεκινήσει η διαδικασία **EraseFash**, ελευθερώνουμε το κουμπί BOOT της πλακέτας και αναμένουμε λίγα δευτερόλεπτα έως ότου ολοκληρωθεί η εγκατάσταση του firmware.

7.1.4.2 Με χρήση του esptool.py

Για να εγκαταστήσουμε το MicroPython firmware μέσω του esptool.py θα πρέπει πρώτα να έχουμε εγκαταστήσει στον υπολογιστή μας την τελευταία έκδοση της Python ακολουθώντας τα βήματα που περιγράψαμε σε προηγούμενη παράγραφο.

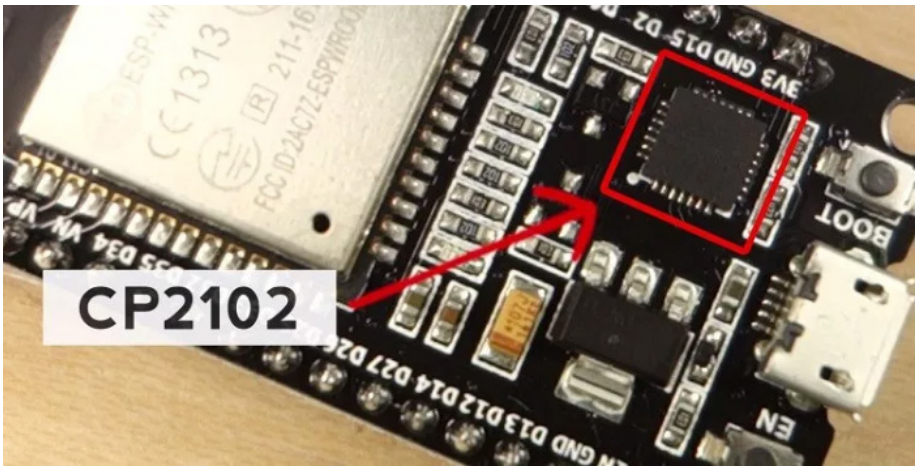
Στη συνέχεια ακολουθούμε την παρακάτω διαδικασία:

- Αρχικά κατεβάζουμε την τελευταία έκδοση του MicroPython firmware για το ESP32, πηγαίνοντας στη σελίδα MicroPython Downloads page[2] και στον τομέα που αφορά τις πλακέτες ESP32.

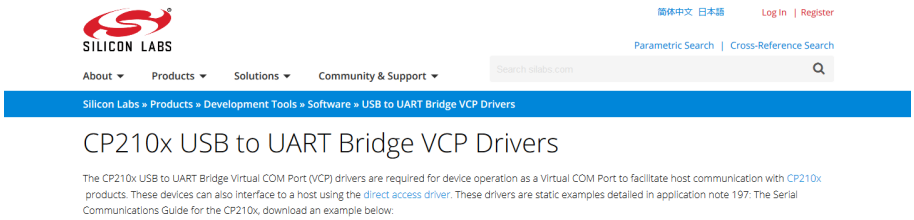


- Συνδέουμε την πλακέτα ESP32 στον υπολογιστή και βρίσκουμε την θύρα COM που έχει συνδεθεί. Αν δεν υπάρχει τότε:
  - Έχουμε χρησιμοποιήσει καλώδιο που δεν έχει δυνατότητα μεταφοράς δεδομένων ή
  - Πρέπει να εγκαταστήσουμε τον απαραίτητο driver για το chipset της πλακέτας. Αυτό γίνεται ως εξής:
    - ✓ βρίσκουμε το όνομα του chipset το οποίο βρίσκεται μπροστά από το κουμπί **BOOT** της πλακέτας

[2] http://micropython.org/download#esp32



✓ Εντοπίζουμε, κατεβάζουμε και εγκαθιστούμε τον σωστό driver. Αν π.χ. το chipset της πλακέτας που χρησιμοποιούμε είναι το CP2102 μπορούμε να πραγματοποιήσουμε τη λήψη μέσω της ιστοσελίδας της Silicon Labs



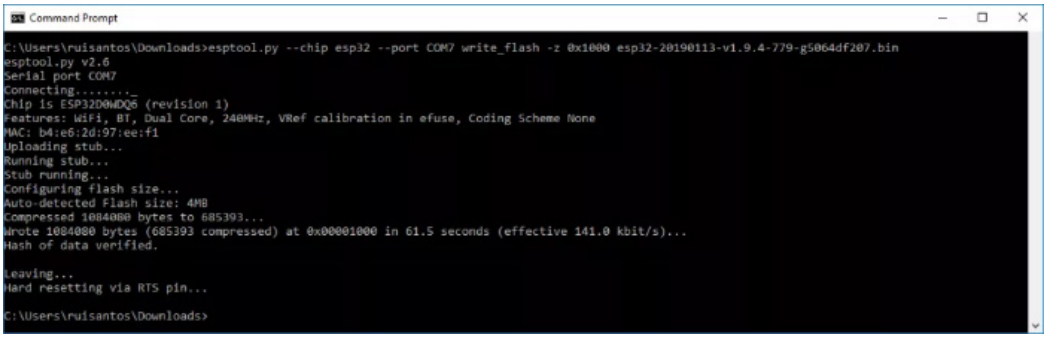
ενώ στη γενική περίπτωση, μπορούμε να εντοπίσουμε τους drivers μέσω κάποιας μηχανής αναζήτησης (π.χ. της Google)

- Ανοίγουμε το Terminal window και εγκαθιστούμε την τελευταία έκδοση του esptool.py γράφοντας: **pip install esptool**
- Πηγαίνουμε στον φάκελο όπου έχουμε κατεβάσει το firmware το οποίο έχει π.χ. το όνομα ***esp32-20190113-v1.9.4-779-g5064df207.bin***
- Έχοντας πατημένο το BOOT στην πλακέτα γράφουμε:
  esptool.py - -chip esp32 erase_flash

- Μόλις η διαδικασία ολοκληρωθεί εάν π.χ. η θύρα συνδεσης της πλακέτας είναι η COM5 εγκαθιστούμε το firmware γράφοντας το ακόλουθο:

  esptool.py - -chip esp32 - -port COM5 write_flash –z 0x1000 esp32-20190113-v1.9.4-779-g5064df207.bin

  και κρατώντας πατημένο το **BOOT** της πλακέτας πατάμε το **ENTER** για τρέξει η εντολή. Μετά από λίγα δευτερόλεπτα θα διαπιστώσουμε ότι η διαδικασία ολοκληρώθηκε βλέποντας μια οθόνη παρόμοια με την ακόλουθη.



7.2 ΠΑΡΑΡΤΗΜΑ – ΠΡΟΓΡΑΜΜΑΤΑ ΟΔΗΓΗΣΗΣ ΣΥΣΤΗΜΑΤΩΝ

7.2.1 Προγράμματα οδήγησης συστήματος ελέγχου της πόρτας γκαράζ

7.2.1.1 boot.py

```
try:
  import usocket as socket
except:
  import socket
from machine import Pin
import network
import time
import machine
import esp
esp.osdebug(None)
import gc
gc.collect()
ssid = 'Network Name'
password = 'password'
station = network.WLAN(network.STA_IF)
station.active(True)
station.ifconfig(('192.###.##.###','255.###.###.0','192.###.##.#','192.###.##.#'))
station.connect(ssid, password)
while station.isconnected() == False:
  pass
print('Connection successful')
print(station.ifconfig())
REL0 = machine.Pin(18, machine.Pin.OUT)
REL2 = machine.Pin(19, machine.Pin.OUT)
rw=machine.Pin(23, machine.Pin.IN, machine.Pin.PULL_UP)
```

```
rw2=machine.Pin(22, machine.Pin.IN, machine.Pin.PULL_UP)
```

7.2.1.2 main.py

```
def web_page():
  if rw.value()==1:
    door="is Open"
  else:
    door="is Closed"

  html = """<html><head> <title>Control Garage's Door</title> <meta http-equiv="refresh"
content="3"> </head>
    <body>
      <h2>Control Garage's Door</h2>
      <br>
      <p>DOOR : <strong>""" + door + """</strong></p>
      <h4>Internet Menu :</h4>
      <form>
          OPTIONS<button name="REL" value="ON0" type="submit">OPEN</button>
                  <button name="REL" value="OFF0" type="submit">CLOSE</button>
                   <button name="REL" value="ON2" type="submit">STOP</button>
        <br>
      </form>
    </body>
  </html>"""
  return html
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('', 80))
s.listen(5)
while True:
  conn, addr = s.accept()
  print("Got a connection from %s" % str(addr))
```

```
request = conn.recv(1024)
print("Content = %s" % str(request))
request = str(request)
RELON0 = request.find('/?REL=ON0')
RELOFF0 = request.find('/?REL=OFF0')
RELON2 = request.find('/?REL=ON2')
RELOFF2 = request.find('/?REL=OFF2')
 if rw.value()== 0:
        print('The Door Stopped')
        REL0.value(0)
        REL2.value(0)
 if rw2.value()== 0:
        print('The Door Stopped')
        REL0.value(0)
        REL2.value(0)
 if ((RELON0 == 6 and (rw.value()==0 or rw.value()==1) and rw2.value()!=0)):
         #if (rw.value()==0 or rw.value()==1):
         print('Now The Door Opens')
         REL0.value(1)
         REL2.value(0)
         time.sleep(4)
 if (RELOFF0 == 6 and rw.value()==1):
         print('Now The Door Closes')
         REL0.value(0)
         REL2.value(1)
         time.sleep(4)
 if RELON2 == 6:
         print('The Door Stopped')
         REL0.value(0)
         REL2.value(0)
response = web_page()
```

```
conn.sendall(response)
conn.close()
```

7.2.2 Προγράμματα οδήγησης συστήματος ελέγχου του φωτισμού δωμάτιου

7.2.2.1 boot.py

```
try:
  import usocket as socket
except:
  import socket
from machine import Pin
import network
import time
import machine
import esp
esp.osdebug(None)
import gc
gc.collect()
ssid = 'Network Name'
password = 'password'
station = network.WLAN(network.STA_IF)
station.active(True)
station.connect(ssid, password)
while station.isconnected() == False:
  pass
print('Connection successful')
print(station.ifconfig())
REL = machine.Pin(19, machine.Pin.OUT)
rw=machine.Pin(23, machine.Pin.IN, machine.Pin.PULL_UP)
```

7.2.2.2 main.py

```
def web_page():
    if rw.value()==1:
```

```
    lights="are CLOSED"
  else:
    lights="are OPEN"

  html = """<html><head> <title>Room's Lights</title> <meta http-equiv="refresh" content="2"> </head>
    <body>
      <h2>Room's Lights</h2>
      <p>LIGHTS : <strong>""" + lights + """</strong></p>
      <br><br>
      <form>
            Internet Menu:<button name="REL" value="ON2" type="submit">OPTION 1
                  </button>
                <button name="REL" value="OFF2" type="submit">OPTION 2</button>
      </form>
    </body>
  </html>"""
  return html

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('', 80))
s.listen(5)

while True:
  conn, addr = s.accept()
  print("Got a connection from %s" % str(addr))
  request = conn.recv(1024)
  print("Content = %s" % str(request))
  request = str(request)
  RELON = request.find('/?REL=ON2')
  RELOFF = request.find('/?REL=OFF2')
```

```
if RELON == 6:
    print('TURN REL2 ON')
    REL.value(1)
if RELOFF == 6:
    print('TURN REL2 OFF')
    REL.value(0)
response = web_page()
conn.sendall(response)
conn.close()
```

7.2.3 Προγράμματα οδήγησης συστήματος ελέγχου του χώρου

7.2.3.1 esp32-cam.ino

```
#include "esp_camera.h"
#include <WiFi.h>
#include "esp_timer.h"
#include "img_converters.h"
#include "Arduino.h"
#include "fb_gfx.h"
#include "soc/soc.h" //disable brownout problems
#include "soc/rtc_cntl_reg.h"  //disable brownout problems
#include "esp_http_server.h"

//Replace with your network credentials
const char* ssid = "*******";
const char* password = "*******";

#define PART_BOUNDARY "123456789000000000000987654321"

// This project was tested with the AI Thinker Model, M5STACK PSRAM Model and
M5STACK WITHOUT PSRAM
#define CAMERA_MODEL_AI_THINKER
//#define CAMERA_MODEL_M5STACK_PSRAM
//#define CAMERA_MODEL_M5STACK_WITHOUT_PSRAM
// Not tested with this model
//#define CAMERA_MODEL_WROVER_KIT

#if defined(CAMERA_MODEL_WROVER_KIT)
  #define PWDN_GPIO_NUM    -1
  #define RESET_GPIO_NUM   -1
  #define XCLK_GPIO_NUM    21
```

```
  #define SIOD_GPIO_NUM    26
  #define SIOC_GPIO_NUM    27
  #define Y9_GPIO_NUM      35
  #define Y8_GPIO_NUM      34
  #define Y7_GPIO_NUM      39
  #define Y6_GPIO_NUM      36
  #define Y5_GPIO_NUM      19
  #define Y4_GPIO_NUM      18
  #define Y3_GPIO_NUM       5
  #define Y2_GPIO_NUM       4
  #define VSYNC_GPIO_NUM   25
  #define HREF_GPIO_NUM    23
  #define PCLK_GPIO_NUM    22

#elif defined(CAMERA_MODEL_M5STACK_PSRAM)
  #define PWDN_GPIO_NUM     -1
  #define RESET_GPIO_NUM    15
  #define XCLK_GPIO_NUM     27
  #define SIOD_GPIO_NUM     25
  #define SIOC_GPIO_NUM     23
  #define Y9_GPIO_NUM       19
  #define Y8_GPIO_NUM       36
  #define Y7_GPIO_NUM       18
  #define Y6_GPIO_NUM       39
  #define Y5_GPIO_NUM        5
  #define Y4_GPIO_NUM       34
  #define Y3_GPIO_NUM       35
  #define Y2_GPIO_NUM       32
  #define VSYNC_GPIO_NUM    22
  #define HREF_GPIO_NUM     26
  #define PCLK_GPIO_NUM     21
```

```
#elif defined(CAMERA_MODEL_M5STACK_WITHOUT_PSRAM)
  #define PWDN_GPIO_NUM     -1
  #define RESET_GPIO_NUM    15
  #define XCLK_GPIO_NUM     27
  #define SIOD_GPIO_NUM     25
  #define SIOC_GPIO_NUM     23
  #define Y9_GPIO_NUM       19
  #define Y8_GPIO_NUM       36
  #define Y7_GPIO_NUM       18
  #define Y6_GPIO_NUM       39
  #define Y5_GPIO_NUM        5
  #define Y4_GPIO_NUM       34
  #define Y3_GPIO_NUM       35
  #define Y2_GPIO_NUM       17
  #define VSYNC_GPIO_NUM    22
  #define HREF_GPIO_NUM     26
  #define PCLK_GPIO_NUM     21

#elif defined(CAMERA_MODEL_AI_THINKER)
  #define PWDN_GPIO_NUM     32
  #define RESET_GPIO_NUM    -1
  #define XCLK_GPIO_NUM      0
  #define SIOD_GPIO_NUM     26
  #define SIOC_GPIO_NUM     27

  #define Y9_GPIO_NUM       35
  #define Y8_GPIO_NUM       34
  #define Y7_GPIO_NUM       39
  #define Y6_GPIO_NUM       36
  #define Y5_GPIO_NUM       21
```

```
  #define Y4_GPIO_NUM       19
  #define Y3_GPIO_NUM       18
  #define Y2_GPIO_NUM        5
  #define VSYNC_GPIO_NUM    25
  #define HREF_GPIO_NUM     23
  #define PCLK_GPIO_NUM     22
#else
  #error "Camera model not selected"
#endif

static const char* _STREAM_CONTENT_TYPE = "multipart/x-mixed-replace;boundary="
PART_BOUNDARY;
static const char* _STREAM_BOUNDARY = "\r\n--" PART_BOUNDARY "\r\n";
static const char* _STREAM_PART = "Content-Type: image/jpeg\r\nContent-Length:
%u\r\n\r\n";
httpd_handle_t stream_httpd = NULL;
static esp_err_t stream_handler(httpd_req_t *req){
  camera_fb_t * fb = NULL;
  esp_err_t res = ESP_OK;
  size_t _jpg_buf_len = 0;
  uint8_t * _jpg_buf = NULL;
  char * part_buf[64];
  res = httpd_resp_set_type(req, _STREAM_CONTENT_TYPE);
  if(res != ESP_OK){
    return res;
  }
  while(true){
    fb = esp_camera_fb_get();
    if (!fb) {
      Serial.println("Camera capture failed");
      res = ESP_FAIL;
```

```
    } else {
      if(fb->width > 400){
        if(fb->format != PIXFORMAT_JPEG){
          bool jpeg_converted = frame2jpg(fb, 80, &_jpg_buf, &_jpg_buf_len);
          esp_camera_fb_return(fb);
          fb = NULL;
          if(!jpeg_converted){
            Serial.println("JPEG compression failed");
            res = ESP_FAIL;
          }
        } else {
          _jpg_buf_len = fb->len;
          _jpg_buf = fb->buf;
        }
      }
    }
    if(res == ESP_OK){
      size_t hlen = snprintf((char *)part_buf, 64, _STREAM_PART, _jpg_buf_len);
      res = httpd_resp_send_chunk(req, (const char *)part_buf, hlen);
    }
    if(res == ESP_OK){
      res = httpd_resp_send_chunk(req, (const char *)_jpg_buf, _jpg_buf_len);
    }
    if(res == ESP_OK){
      res = httpd_resp_send_chunk(req, _STREAM_BOUNDARY,
strlen(_STREAM_BOUNDARY));
    }
    if(fb){
      esp_camera_fb_return(fb);
      fb = NULL;
      _jpg_buf = NULL;
```

```
    } else if(_jpg_buf){
      free(_jpg_buf);
      _jpg_buf = NULL;
    }
    if(res != ESP_OK){
      break;
    }
    //Serial.printf("MJPG: %uB\n",(uint32_t)(_jpg_buf_len));
  }
  return res;
}

void startCameraServer(){
  httpd_config_t config = HTTPD_DEFAULT_CONFIG();
  config.server_port = 80;

  httpd_uri_t index_uri = {
    .uri       = "/",
    .method    = HTTP_GET,
    .handler   = stream_handler,
    .user_ctx  = NULL
  };

  //Serial.printf("Starting web server on port: '%d'\n", config.server_port);
  if (httpd_start(&stream_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(stream_httpd, &index_uri);
  }
}

void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); //disable brownout detector
```

```
Serial.begin(115200);
Serial.setDebugOutput(false);
camera_config_t config;
config.ledc_channel = LEDC_CHANNEL_0;
config.ledc_timer = LEDC_TIMER_0;
config.pin_d0 = Y2_GPIO_NUM;
config.pin_d1 = Y3_GPIO_NUM;
config.pin_d2 = Y4_GPIO_NUM;
config.pin_d3 = Y5_GPIO_NUM;
config.pin_d4 = Y6_GPIO_NUM;
config.pin_d5 = Y7_GPIO_NUM;
config.pin_d6 = Y8_GPIO_NUM;
config.pin_d7 = Y9_GPIO_NUM;
config.pin_xclk = XCLK_GPIO_NUM;
config.pin_pclk = PCLK_GPIO_NUM;
config.pin_vsync = VSYNC_GPIO_NUM;
config.pin_href = HREF_GPIO_NUM;
config.pin_sscb_sda = SIOD_GPIO_NUM;
config.pin_sscb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 20000000;
config.pixel_format = PIXFORMAT_JPEG;


if(psramFound()){
  config.frame_size = FRAMESIZE_UXGA;
  config.jpeg_quality = 10;
  config.fb_count = 2;
} else {
```

```
  config.frame_size = FRAMESIZE_SVGA;
  config.jpeg_quality = 12;
  config.fb_count = 1;
 }
 // Camera init
 esp_err_t err = esp_camera_init(&config);
 if (err != ESP_OK) {
  Serial.printf("Camera init failed with error 0x%x", err);
  return;
 }
 // Wi-Fi connection
 WiFi.begin(ssid, password);
 while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
 }
 Serial.println("");
 Serial.println("WiFi connected");
 Serial.print("Camera Stream Ready! Go to: http://");
 Serial.print(WiFi.localIP());
 // Start streaming web server
 startCameraServer();
}
void loop() {
 delay(1);
}
```

7.2.3.2 theft_alert.ino

```
#include <HTTPClient.h>
#include <WiFi.h>
#include "esp_http_server.h"
```

```
#include "esp_camera.h"
#include "Arduino.h"
#include "FS.h"                // SD Card ESP32
#include "SD_MMC.h"            // SD Card ESP32
#include "soc/soc.h"           // Disable brownour problems
#include "soc/rtc_cntl_reg.h"  // Disable brownour problems
#include "driver/rtc_io.h"
#include <EEPROM.h>            // read and write from flash memory
// define the number of bytes you want to access
#define EEPROM_SIZE 1

RTC_DATA_ATTR int bootCount = 0;

// Pin definition for CAMERA_MODEL_AI_THINKER
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27
#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22
```

```
int pictureNumber = 0;

#define uS_TO_S_FACTOR 1000000

const char* ssid = "**************";
const char* password = "**************";

void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); //disable brownout detector
  Serial.begin(115200);

  Serial.setDebugOutput(true);

  setupWifi();

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
```

```
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;

  pinMode(4, INPUT);
  digitalWrite(4, LOW);
  rtc_gpio_hold_dis(GPIO_NUM_4);

  if(psramFound()){
    config.frame_size = FRAMESIZE_UXGA; // FRAMESIZE_ +
QVGA|CIF|VGA|SVGA|XGA|SXGA|UXGA
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }

  // Init Camera
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }
   Serial.println("Starting SD Card");

  delay(500);
  if(!SD_MMC.begin()){
```

```
  Serial.println("SD Card Mount Failed");
  //return;
}

uint8_t cardType = SD_MMC.cardType();
if(cardType == CARD_NONE){
  Serial.println("No SD Card attached");
  return;
}

camera_fb_t * fb = NULL;

// Take Picture with Camera
fb = esp_camera_fb_get();
if(!fb) {
  Serial.println("Camera capture failed");
  return;
}
// initialize EEPROM with predefined size
EEPROM.begin(EEPROM_SIZE);
pictureNumber = EEPROM.read(0) + 1;

// Path where new picture will be saved in SD Card
String path = "/picture" + String(pictureNumber) +".jpg";

fs::FS &fs = SD_MMC;
Serial.printf("Picture file name: %s\n", path.c_str());

File file = fs.open(path.c_str(), FILE_WRITE);
if(!file){
  Serial.println("Failed to open file in writing mode");
```

```
}
else {
  file.write(fb->buf, fb->len); // payload (image), payload length
  Serial.printf("Saved file to path: %s\n", path.c_str());
  EEPROM.write(0, pictureNumber);
  EEPROM.commit();

  if ((pictureNumber !=0))
     {
       Serial.println("ALERT!!");
       while (get_http(String("Alert")) != 0);

     }
  delay(5000);
}

file.close();
esp_camera_fb_return(fb);

delay(1000);

// Turns off the ESP32-CAM white on-board LED (flash) connected to GPIO 4
pinMode(4, OUTPUT);
digitalWrite(4, LOW);
rtc_gpio_hold_en(GPIO_NUM_4);

esp_sleep_enable_ext0_wakeup(GPIO_NUM_13, 0);
 Serial.println("Going to sleep now");
delay(1000);
esp_deep_sleep_start();
Serial.println("This will never be printed");
```

```
}

void loop() {

}

void setupWifi()
{
    WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
}

int get_http(String state)
{
   HTTPClient http;
   int ret = 0;
   Serial.print("[HTTP] begin...\n");
   // configure ifttt server and url  should be HTTP only..not https!!!  (http://)
    http.begin("http://maker.ifttt.com/trigger/Alert/with/key/p-
r4FoVRksK_2QgBQQs7eKVssZPtwCq2I7zBk_6yyBN"); //HTTP


    Serial.print("[HTTP] GET...\n");
    // start connection and send HTTP header
    int httpCode = http.GET();
    // httpCode will be negative on error
```

```
    if(httpCode > 0) {
    // HTTP header has been send and Server response header has been handled
    Serial.printf("[HTTP] GET code: %d\n", httpCode);

      if(httpCode == HTTP_CODE_OK) {
        String payload = http.getString();
        Serial.println(payload);
      }
    } else {
        ret = -1;
        Serial.printf("[HTTP] GET failed, error: %s\n", http.errorToString(httpCode).c_str());
        delay(3000); // wait for three sec before retry again
    }

    http.end();
    return ret;
}
```

7.2.4 Προγράμματα οδήγησης συστήματος ελέγχου του κλιματισμού

7.2.4.1 boot.py

```
try:
  import usocket as socket
except:
  import socket
from machine import Pin, ADC
import network
import time
import machine
import dht
import esp
esp.osdebug(None)
import gc
gc.collect()
ssid = 'Network Name'
password = 'password'
station = network.WLAN(network.STA_IF)
station.active(True)
station.connect(ssid, password)
while station.isconnected() == False:
  pass
print('Connection successful')
print(station.ifconfig())
REL = machine.Pin(19, machine.Pin.OUT)
REL.value(1)
sensor = dht.DHT11(Pin(4))
pot = ADC(Pin(34))
pot.atten(ADC.ATTN_11DB)
```

7.2.4.2 main.py

```
def read_sensor():
  global temp, hum
  temp = hum = 0
  try:
    sensor.measure()
    temp = sensor.temperature()
    hum = sensor.humidity()
    if (isinstance(temp, float) and isinstance(hum, float)) or (isinstance(temp, int) and isinstance(hum, int)):
      msg = (b'{0:3.1f},{1:3.1f}'.format(temp, hum))

      hum = round(hum, 2)
      return(msg)
    else:
      return('Invalid sensor readings.')
  except OSError as e:
    return('Failed to read sensor.')

def web_page():
    if pot.read()==0:
       air="is CLOSED"
    else:
       air="is OPEN"

    html = """<html><head> <title>Control Clima</title> <meta http-equiv="refresh" content="3"> </head>
        <body>
           <h2>Control CLIMA</h2>
           <p>
              <span> Temperature : </span>
```

```
            <span>"""+str(temp)+"""</span>
            <sup class="units">&deg;C</sup>
            <span> Humidity :</span>
            <span>"""+str(hum)+"""</span>
            <sup class="units">%</sup>
          </p>
          <br><br>
          <p>Air Condition : <strong>""" + air + """</strong></p>
          <br><br>
              <form>
           Internet Menu:<button name="REL" value="OFF2" type="submit">OPTION 1
                          </button>
            <button name="REL" value="ON2" type="submit">OPTION 2</button>
        </form>
       </body>
    </html>"""
    return html
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('', 80))
s.listen(5)

while True:
    conn, addr = s.accept()
    print("Got a connection from %s" % str(addr))
    request = conn.recv(1024)
    print("Content = %s" % str(request))
    request = str(request)
    RELON = request.find('/?REL=ON2')
    RELOFF = request.find('/?REL=OFF2')
    if RELON == 6:
        print('TURN REL2 ON')
```

```
    REL.value(1)
if RELOFF == 6:
    print('TURN REL2 OFF')
    REL.value(0)
sensor_readings = read_sensor()
print(sensor_readings)
response = web_page()
conn.sendall(response)
conn.close()
```

7.2.5 Προγράμματα οδήγησης συστήματος ελέγχου τιμών περιβαλλοντικών μεταβλητών

7.2.5.1 bmp180.py

```
from ustruct import unpack as unp
from machine import I2C, Pin
import math
import time

# BMP180 class
class BMP180():
    '''
    Module for the BMP180 pressure sensor.
    '''

    _bmp_addr = 119             # adress of BMP180 is hardcoded on the sensor

    # init
    def __init__(self, i2c_bus):

        # create i2c obect
        _bmp_addr = self._bmp_addr
        self._bmp_i2c = i2c_bus
        self._bmp_i2c.start()
        self.chip_id = self._bmp_i2c.readfrom_mem(_bmp_addr, 0xD0, 2)
        # read calibration data from EEPROM
        self._AC1 = unp('>h', self._bmp_i2c.readfrom_mem(_bmp_addr, 0xAA, 2))[0]
        self._AC2 = unp('>h', self._bmp_i2c.readfrom_mem(_bmp_addr, 0xAC, 2))[0]
        self._AC3 = unp('>h', self._bmp_i2c.readfrom_mem(_bmp_addr, 0xAE, 2))[0]
        self._AC4 = unp('>H', self._bmp_i2c.readfrom_mem(_bmp_addr, 0xB0, 2))[0]
        self._AC5 = unp('>H', self._bmp_i2c.readfrom_mem(_bmp_addr, 0xB2, 2))[0]
```

```
    self._AC6 = unp('>H', self._bmp_i2c.readfrom_mem(_bmp_addr, 0xB4, 2))[0]
    self._B1 = unp('>h', self._bmp_i2c.readfrom_mem(_bmp_addr, 0xB6, 2))[0]
    self._B2 = unp('>h', self._bmp_i2c.readfrom_mem(_bmp_addr, 0xB8, 2))[0]
    self._MB = unp('>h', self._bmp_i2c.readfrom_mem(_bmp_addr, 0xBA, 2))[0]
    self._MC = unp('>h', self._bmp_i2c.readfrom_mem(_bmp_addr, 0xBC, 2))[0]
    self._MD = unp('>h', self._bmp_i2c.readfrom_mem(_bmp_addr, 0xBE, 2))[0]

    # settings to be adjusted by user
    self.oversample_setting = 3
    self.baseline = 101325.0

    # output raw
    self.UT_raw = None
    self.B5_raw = None
    self.MSB_raw = None
    self.LSB_raw = None
    self.XLSB_raw = None
    self.gauge = self.makegauge() # Generator instance
    for _ in range(128):
        next(self.gauge)
        time.sleep_ms(1)

def compvaldump(self):
    '''
    Returns a list of all compensation values
    '''
    return [self._AC1, self._AC2, self._AC3, self._AC4, self._AC5, self._AC6,
            self._B1, self._B2, self._MB, self._MC, self._MD, self.oversample_setting]

# gauge raw
def makegauge(self):
```

```
        '''
        Generator refreshing the raw measurments.
        '''
        delays = (5, 8, 14, 25)
        while True:
            self._bmp_i2c.writeto_mem(self._bmp_addr, 0xF4, bytearray([0x2E]))
            t_start = time.ticks_ms()
            while (time.ticks_ms() - t_start) <= 5: # 5mS delay
                yield None
            try:
                self.UT_raw = self._bmp_i2c.readfrom_mem(self._bmp_addr, 0xF6, 2)
            except:
                yield None
            self._bmp_i2c.writeto_mem(self._bmp_addr, 0xF4,
bytearray([0x34+(self.oversample_setting << 6)]))
            t_pressure_ready = delays[self.oversample_setting]
            t_start = time.ticks_ms()
            while (time.ticks_ms() - t_start) <= t_pressure_ready:
                yield None
            try:
                self.MSB_raw = self._bmp_i2c.readfrom_mem(self._bmp_addr, 0xF6, 1)
                self.LSB_raw = self._bmp_i2c.readfrom_mem(self._bmp_addr, 0xF7, 1)
                self.XLSB_raw = self._bmp_i2c.readfrom_mem(self._bmp_addr, 0xF8, 1)
            except:
                yield None
            yield True

    def blocking_read(self):
        if next(self.gauge) is not None: # Discard old data
            pass
        while next(self.gauge) is None:
```

```
        pass

    @property
    def oversample_sett(self):
        return self.oversample_setting

    @oversample_sett.setter
    def oversample_sett(self, value):
        if value in range(4):
            self.oversample_setting = value
        else:
            print('oversample_sett can only be 0, 1, 2 or 3, using 3 instead')
            self.oversample_setting = 3

    @property
    def temperature(self):
        '''
        Temperature in degree C.
        '''
        next(self.gauge)
        try:
            UT = unp('>H', self.UT_raw)[0]
        except:
            return 0.0
        X1 = (UT-self._AC6)*self._AC5/2**15
        X2 = self._MC*2**11/(X1+self._MD)
        self.B5_raw = X1+X2
        return (((X1+X2)+8)/2**4)/10

    @property
    def pressure(self):
```

```
'''
Pressure in mbar.
'''
next(self.gauge)
self.temperature  # Populate self.B5_raw
try:
   MSB = unp('B', self.MSB_raw)[0]
   LSB = unp('B', self.LSB_raw)[0]
   XLSB = unp('B', self.XLSB_raw)[0]
except:
   return 0.0
UP = ((MSB << 16)+(LSB << 8)+XLSB) >> (8-self.oversample_setting)
B6 = self.B5_raw-4000
X1 = (self._B2*(B6**2/2**12))/2**11
X2 = self._AC2*B6/2**11
X3 = X1+X2
B3 = ((int((self._AC1*4+X3)) << self.oversample_setting)+2)/4
X1 = self._AC3*B6/2**13
X2 = (self._B1*(B6**2/2**12))/2**16
X3 = ((X1+X2)+2)/2**2
B4 = abs(self._AC4)*(X3+32768)/2**15
B7 = (abs(UP)-B3) * (50000 >> self.oversample_setting)
if B7 < 0x80000000:
   pressure = (B7*2)/B4
else:
   pressure = (B7/B4)*2
X1 = (pressure/2**8)**2
X1 = (X1*3038)/2**16
X2 = (-7357*pressure)/2**16
return pressure+(X1+X2+3791)/2**4
```

7.2.5.2 boot.py

```
try:
  import usocket as socket
except:
  import socket

import network

from bmp180 import BMP180
from machine import I2C, Pin
from machine import Pin
import dht
import esp
esp.osdebug(None)

import gc
gc.collect()

ssid = 'Network Name'
password = 'password'

station = network.WLAN(network.STA_IF)

station.active(True)
station.connect(ssid, password)

while station.isconnected() == False:
  pass

print('Connection successful')
print(station.ifconfig())
```

```
sensor = dht.DHT22(Pin(4))


bus =  I2C(scl=Pin(22), sda=Pin(21), freq=100000)
bmp180 = BMP180(bus)
```

7.2.5.3 main.py

```
def read_bmp180():
  global pres, alti
  pres = alti = 0
  try:
    bmp180 = BMP180(bus)
    bmp180.oversample_sett = 2
    bmp180.baseline = 101325
    pres = bmp180.pressure
    if (isinstance(pres, float) and isinstance(alti, float)) or (isinstance(alti, int) and isinstance(pres,
int)):
      msg = (b'{0:3.1f},{1:3.1f}'.format(pres, alti))


      return(msg)
    else:
      return('Invalid sensor readings.')
  except OSError as e:
    return('Failed to read sensor.')

def read_sensor():
  global temp, hum
  temp = hum = 0
  try:
    sensor.measure()
    temp = sensor.temperature()
```

```
    hum = sensor.humidity()-25
    if (isinstance(temp, float) and isinstance(hum, float)) or (isinstance(temp, int) and isinstance(hum, int)):
      msg = (b'{0:3.1f},{1:3.1f}'.format(temp, hum))
      hum = round(hum, 2)
      return(msg)
    else:
      return('Invalid sensor readings.')
  except OSError as e:
    return('Failed to read sensor.')

def web_page():
  html = """<!DOCTYPE HTML><html>
    <head>
      <meta charset= "UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1">
      <meta http-equiv="refresh" content="3">
      <style>
        html {
          font-family: Arial;
          display: inline-block;
          margin: 0px auto;
          text-align: center;
        }
        h2 { font-size: 2.0rem; }
        p { font-size: 2.0rem; }
        .units { font-size: 1rem; }
        .labels{
          font-size: 1rem;
          vertical-align:middle;
          padding-bottom: 15px;
```

```
        }
      </style>
    </head>
    <body>
      <h2>Περιβαλλοντικές ενδείξεις</h2>
      <p>
        <span class="labels">Temperature</span>
        <span>"""+str(temp)+"""</span>
        <sup class="units">&deg;C</sup>
        <span class="labels">Humidity</span>
        <span>"""+str(hum)+"""</span>
        <sup class="units">%</sup>
        <span class="labels">Pressure</span>
        <span>"""+str(pres)+"""</span>
        <sup class="units">Pa</sup>
      </p>
    </body>
</html>"""
  return html

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('', 80))
s.listen(5)

while True:
  conn, addr = s.accept()
  print('Got a connection from %s' % str(addr))
  request = conn.recv(1024)
  print('Content = %s' % str(request))
  sensor_readings = read_sensor()
  print(sensor_readings)
```

```
bmp180_reading = read_bmp180()
print(bmp180_reading)
response = web_page()
conn.send('HTTP/1.1 200 OK\n')
conn.send('Content-Type: text/html\n')
conn.send('Connection: close\n\n')
conn.sendall(response)
conn.close()
```

7.2.6 Κώδικας ιστοσελίδας του κέντρου ελέγχου των υποσυστημάτων

```
<!DOCTYPE html>
<html>
	<head>
		<style>
		.flex-container {
		  display: -webkit-flex;
		  display: flex;
		  -webkit-flex-flow: row wrap;
		  flex-flow: row wrap;
		  font-weight: bold;
		  text-align: center;
		}

		.flex-container > * {
		  padding: 3px;
		  flex: 1 100%;
		}

				.header {background: lightgray;}
		.footer {background: white;}
		.aside1 {background: lightgray;}
		.aside2 {background: lightgray;}
		.aside3 {background: lightgray;}
		.aside4 {background: lightgray;}

		@media all and (min-width: 768px) {
			.aside { flex: 1 auto; }
			.aside1 { order: 1; }
			.aside2 { order: 2; }
			.aside3 { order: 3; }
```

```
                .aside4 { order: 4; }
                .footer  { order: 5; }
        }
        </style>
        <title>Smart Home Control Panel</title>
</head>

<body>
        <h1 align='center'> Smart Home Control Panel </h1>
        <div class="flex-container"height="100%" width="800">
                <header class="header"><iframe src="http://***.***.**.**" height="150"
                        width="100%"> METEO STASION </iframe></header>
                <aside class="aside aside1"><iframe src="http:// ***.***.**.**"
                        height="300" width="300"> Air Condition </iframe></aside>
                <aside class="aside aside2"><iframe src="http:// ***.***.**.**"
                        height="300" width="300"> Lights </iframe></aside>
                <aside class="aside aside3"><iframe src="http:// ***.***.**.**"
                        height="300" width="300"> Garage </iframe></aside>
                <aside class="aside aside4"><iframe src="http:// ***.***.**.**"
                        height="300" width="300"> Alarm </iframe></aside>
                <footer class="footer">
                        <h3>Diplomatiki Ergasia Assimakopoulou Fotiou</h3>
                </footer>
        </div>

</body>

</html>
```