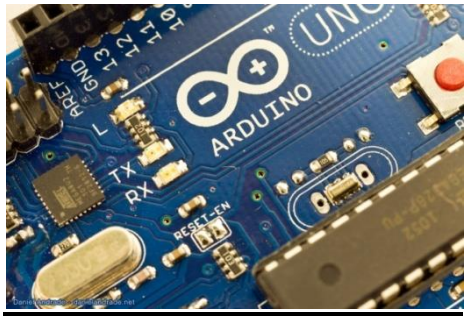
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

ΣΧΟΛΗ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Πτυχιακή Εργασία



Σχεδιασμός και ανάπτυξη «έξυπνης» μονάδας μετρήσεων με την χρήση της πλατφόρμας Arduino.

Χασουράκης Στέφανος, Α.Μ. i68

Επιβλέπων Καθηγητής: Βαρτζιώτης Φώτιος

Άρτα, Ιούλιος, 2019

ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

ΣΧΟΛΗ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Πτυχιακή Εργασία

Σχεδιασμός και ανάπτυξη «έξυπνης» μονάδας μετρήσεων με την χρήση της πλατφόρμας Arduino.

Χασουράκης Στέφανος, Α.Μ. i68

Επιβλέπων Καθηγητής: Βαρτζιώτης Φώτιος

Άρτα, Ιούλιος, 2019

DESING AND DEVELOPMENT OF SMART MEASURING UNIT USING ARDUINO PLATFORM

Εγκρίθηκε από τριμελή εξεταστική επιτροπή

Άρτα, 2019

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Επιβλέπων καθηγητής
   Όνομα: Φώτιος
   Επίθετο: Βαρτζιώτης
   Βαθμίδα: Επίκουρος καθηγητής

2. Μέλος επιτροπής
   Όνομα:
   Επίθετο:
   Βαθμίδα

3. Μέλος επιτροπής
   Όνομα:
   Επίθετο:
   Βαθμίδα:

Ο/Η Προϊστάμενος/η του Τμήματος
Όνομα:
Επίθετο:
Βαθμίδα:

Υπογραφή

© Χασουράκης, Στέφανος ,2019.

Με επιφύλαξη παντός **δικαιώματος**. Allrightsreserved.

Δήλωση μη λογοκλοπής

Δηλώνω υπεύθυνα και γνωρίζοντας τις κυρώσεις του Ν. 2121/1993 περί Πνευματικής Ιδιοκτησίας, ότι η παρούσα πτυχιακή εργασία είναι εξ ολοκλήρου αποτέλεσμα δικής μου ερευνητικής εργασίας, δεν αποτελεί προϊόν αντιγραφής ούτε προέρχεται από ανάθεση σε τρίτους. Όλες οι πηγές που χρησιμοποιήθηκαν (κάθε είδους, μορφής και προέλευσης) για τη συγγραφή της περιλαμβάνονται στη βιβλιογραφία.

Χασουράκης Στέφανος

Υπογραφή

ΕΥΧΑΡΙΣΤΙΕΣ

Με το πέρας της παρούσας πτυχιακής εργασίας θα ήθελα να πω ένα μεγάλο ευχαριστώ σε όλους όσους με στήριξαν από το περιβάλλον μου αλλά και ένα ιδιαίτερο ευχαριστώ στον επιβλέποντα καθηγητή μου Κ. Βαρτζιώτη Φώτιο που με την πολύτιμη βοήθεια του αλλά και την συνεχή και σωστή καθοδήγηση του δημιουργήθηκε η πτυχιακή μου εργασία.

ΠΕΡΙΛΗΨΗ

Η πλατφόρμα που υλοποιήθηκε στην παρούσα εργασία έχει ως σκοπό την άμεση και έγκυρη ενημέρωση κάθε χρήστη για τις καιρικές συνθήκες που επικρατούν καθώς και το φορτίο μιας μονάδας ενδιαφέροντος στο σημείο όπου θα τοποθετηθεί η έξυπνη μονάδα μετρήσεων χωρίς να είναι απαραίτητη η φυσική παρουσία στον περιβάλλοντα χώρο. Η μονάδα είναι βασισμένη στον μικροελεγκτή Arduino Uno R3 ο οποίος μαζί με την βοήθεια του Gsm Shield 2 και των αισθητήρων DHT11 (θερμοκρασίας/υγρασίας) και του load cell για την μέτρηση του βάρους προγραμματίστηκαν έτσι ώστε να μπορεί να μεταδώσει όλα τα δεδομένα που συλλέγει και να απεικονίζονται σε μια ιστοσελίδα.

Υλοποιήθηκε και μια σελίδα για την προβολή των αποτελεσμάτων με γραφήματα με τους μέσους όρους και την τάση για κάθε στοιχείο μέτρησης και έχει δημιουργηθεί ένας πίνακας ο οποίος περιέχει όλες τις μετρήσεις που έχουν πραγματοποιηθεί.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ

ArduinoUno, μικροελεγκτής, προγραμματισμός, php, MySQL, Γλώσσα C, αισθητήρες, μετάδοση δεδομένων, GSM δίκτυο,

ABSTRACT

In this essay it is described the implementation of a platform that aims to inform each user for the weather conditions and the weight of an object in a specific place where the platform will be set without the user had to be there to have all this information. The platform is created with the use of Arduino Uno R3 and with some modules and sensors. The arduino has been programmed to send all the data to a website.

The web site processes and displays the transmitted sensor-oriented data. Specifically, it includes a table that presents in real time the data measured by the selected sensors, and four graphs that depict the trend of the transmitted data in hourly, weekly, monthly and yearly basis.

KEYWORDS

Arduino Uno, microcontroller , programming, Php, MySQL, C Language, sensors, data transmition, GSM network

Περιεχόμενα

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

1. Εισαγωγή

Το Internet of Things (IoT) θεωρείται είναι μία από τις τρεις κορυφαίες τεχνολογικές εξελίξεις της δεκαετίας (mobileInternet και την αυτοματοποίηση του knowledgework).[1] Το IoT είναι συνδεδεμένο με μια τεράστια γκάμα προϊόντων , συστημάτων ,αισθητήρων. Ο σωστός συνδυασμός όλων των παραπάνω μπορούν να παράγουν δυνατότητες οι οποίες είναι πέραν τον προσδοκιών όπου με έναν τρόπο θα είναι ικανά να επηρεάσουν την καθημερινότητα των ανθρώπων. Το IoT είναι μια από τις κορυφαίες τεχνολογικές εξελίξεις της εικοσαετίας. Κύριος σκοπός του IoT είναι μέσω συσκευών ή συστημάτων όπου μπορούν να προγραμματιστούν και να ελεγχθούν να παράγονται πληροφορίες οι οποίες θα διευκολύνουν τον τρόπο ζωής , θα δημιουργήσουν νέες υπηρεσίες και ευκαιρίες και μελλοντικά να βοηθήσουν στην προστασία του περιβάλλοντος. Αναμφίβολα το IoT πραγματοποιεί μια ραγδαία εξέλιξη τα τελευταία χρόνια και αναμένεται ακόμα μεγαλύτερη.

Μια από τις συσκευές η οποία είναι συσχετισμένη με το IoT είναι ο μικροεπεξεργαστής Arduino. Το Arduino είναι μια αυτόνομη μητρική πλακέτα ανοιχτού κώδικα, με ενσωματωμένο μικροελεγκτή, με εισόδους-εξόδους όπου μπορεί να προγραμματιστεί με την γλώσσα Wiring [2]. Οι δυνατότητες που παρέχει η συγκεκριμένη πλακέτα είναι πολλές διότι έχει την δυνατότητα παράλληλης σύνδεσης πολλών αντικειμένων input-output που προσφέρουν μια εύκολη και εύχρηστη διεπαφή με τον χρήστη.

Στην παρούσα εργασία σκοπός είναι η δημιουργία ενός «έξυπνου» συστήματος με την βοήθεια της πλακέτας Arduino. Η βασική ιδέα του project γεννήθηκε όταν χρειάστηκε να έχω κάποιες ακριβείς μετρήσεις σε ένα απομακρυσμένο σημείο. Έτσι με την βοήθεια του Arduino και μερικών αισθητήρων δημιουργήθηκε η «έξυπνη μονάδα μετρήσεων» .Στην μονάδα θα ενσωματωθεί και το GSM Module έτσι ώστε να μπορεί γίνεται αναμετάδοση για τις συνθήκες που επικρατούν πραγματοποιώντας μετρήσεις για την θερμοκρασία, υγρασία και το βάρος ενός αντικειμένου. Οι μετρήσεις θα είναι σε Celsius, ποσοστό επί τις εκατό και κιλά αντίστοιχα. Ο τρόπος μετάδοσης των αποτελεσμάτων θα πραγματοποιείται μέσω ενός μηνύματος ή με την χρήση του διαδικτύου όπου τα αποτελέσματα αποστέλλονται σε έναν διακομιστή.

Στα επόμενα κεφάλαια της εργασίας θα παρουσιαστεί αναλυτικά ποια είναι η πλακέτα Arduino, λειτουργίες που παρέχει και το πώς δημιουργήθηκε αναλυτικά το

project. Στο τελευταίο κεφάλαιο θα υπάρχουν τα συμπεράσματα καθώς και η μελλοντική εξέλιξη που θα μπορέσει να πραγματοποιηθεί.

2. Προσδιορίζοντας την «έξυπνη μονάδα μετρήσεων»

Στις αρχές τις δεκαετίας του 1970 ξεκίνησε η αναφορά της λέξης «έξυπνο» σε νέα τεχνολογικά επιτεύγματα. Τα πρώτα προϊόντα τα όποια έφεραν την λέξη «έξυπνο» ήταν οι βόμβες και οι πύραυλοι όπου χρησιμοποιούσε ο Αμερικάνικος στρατός και είχαν το όνομα «έξυπνες βόμβες» οι οποίες είχαν την δυνατότητα να προσδιορίζουν την διαδρομή του εαυτού τους από μόνες τους ως προς τον στόχο. Στην συνέχεια στην δεκαετία του 1980 ο ορισμός ενός έξυπνου προϊόντος ξεκίνησε να χρησιμοποιείται και σε υπολογιστές, οικιακές συσκευές προηγμένης τεχνολογίας αλλά σε μεγαλύτερο βαθμό σε ολοκληρωμένα κυκλώματα. Ο όρος αυτός στην συνέχεια των δεκαετιών σταμάτησε να αποκαλείται σε συσκευές όπως είναι ο ηλεκτρονικός υπολογιστής κ.α. παρόλο που με την πάροδο τον χρόνων έγιναν ακόμα πιο ισχυροί.

Η έξυπνη μονάδα μετρήσεων που θα υλοποιηθεί στην παρούσα εργασία έχει ως σκοπό την απομακρυσμένη συλλογή αποτελεσμάτων (Θερμοκρασίας , υγρασίας, βάρους).Την αυτόματη δημιουργία μια βάσης δεδομένων. Μια ανάλυση των συνθηκών με βάση τον μέσο όρο ανά ημέρα, εβδομάδα , μήνα και χρόνο.Τα αποτελέσματαθα απεικονίζονται με διαγράμματα.

2.1 Τεχνολογία υλοποίησης της έξυπνης μονάδας

Η βασική πλακέτα που θα χρησιμοποιηθεί για την «έξυπνη μονάδα μετρήσεων» είναι το Arduino. Ο λόγος που επιλέχθηκε η συγκεκριμένη είναι διότι παρέχει πολλαπλές χρήσεις με προσιτό οικονομικό κόστος. Μάλιστα οι επεκτάσεις (modules) και οι αισθητήρες οι οποίοι μπορούν να χρησιμοποιηθούν είναι και εκείνοι σε αρκετά προσιτή τιμή.

Ο λόγος που προτιμήθηκε η δημιουργία μιας τέτοιας μονάδας είναι διότι ο κάθε χρήστης θα μπορεί να την τοποθετεί σε οποιοδήποτε απομακρυσμένο σημείο επιθυμεί. Το ιδανικό θα ήταν να δημιουργηθεί κάτι αντίστοιχο με ενσύρματη δικτύωση το οποίο είναι σαφές πιο ασφαλές και σταθερό αλλά περιορίζει το σημείο όπου μπορεί να τοποθετηθεί ενώ παράλληλα θα είναι πιο ακριβό στην υλοποίηση του.

Στην συνέχεια θα γίνει αναφορά στις τεχνολογίες που θα χρησιμοποιηθούν για την πραγματοποίηση της έξυπνης μονάδας. Θα παρουσιαστεί ξεχωριστά κάθε μια με τα πλεονεκτήματα και τα μειονεκτήματα της όπου θα αναλύεται για ποιόν λόγο επιλέχθηκε η

κάθε μια ξεχωριστά διότι μπορεί να μην είναι η ιδανική επιλογή αλλά σίγουρα καλύπτει τις ανάγκες του project που θα δημιουργηθεί.

2.2 **Μικροελεγκτής Arduino**

Το Arduino θεωρείται ένας μικροελεγκτής μονής πλακέτας. Ουσιαστικά ορίζεται ως μια μητρική πλακέτα ανοιχτού κώδικα όπου έχει ενσωματωμένη ένα μικροελεγκτή και διάφορες εισόδους και εξόδους. Η γλώσσα προγραμματισμού του είναι η Wiring η οποία είναι η γλώσσα προγραμματισμού C++ υποβοηθούμενη από ένα σύνολο βιβλιοθηκών οι οποίες είναι υλοποιημένες στην C++[3]. Σχεδόν όλες οι εκδόσεις του αγοράζονται προ-συναρμολογημένες όπου οι πληροφορίες για το διάγραμμα τους και τον τρόπο συναρμολόγησης του είναι διαθέσιμο για όποιον επιθυμεί να το συναρμολογήσει μόνος του. Επίσης όλες οι εκδόσεις αγοράζονται έτοιμες συναρμολογημένες και προγραμματισμένες έτοιμες για χρήση.

Το 2005 γεννήθηκε η ιδέα για την κατασκευή μιας συσκευής η οποία θα είχε τον έλεγχο προγραμμάτων από μαθητές όπου το κόστος αγοράς του θα είναι πιο οικονομικό από τα υπόλοιπα της αγοράς. Ο MassimoBanzi και ο DavidCueartielles είναι οι ιδρυτές του Arduino όπου η κατασκευή του ξεκίνησε σε μια κωμόπολη του Τορίνο στην Ιβρέα της Ιταλίας. Το ονομά του προέρχεται από τον βασιλιά της Ιταλίας Arduin.

ΤοArduino βασίζεται στον μικροελεγκτήATmega328p 8 bitRISC ο οποίος χρονίζεται στα 16MHz. Διαθέτει τρείς διαφορετικές μνήμες.

I. Μνήμη SRAM: 2Kb η οποία χρησιμοποιείται για την αποθήκευση μεταβλητών, πινάκων κλπ κατά το runtime όπου τα δεδομένα χάνονται στην διακοπή παροχής ρεύματος ή εάν πραγματοποιηθεί reset.
II. Μνήμη EEPROM: 1Kb όπου χρησιμοποιείται για την εγγραφή/ανάγνωση δεδομένων κατά το runtime χωρίς να υπάρχει απώλεια των δεδομένων σε περίπτωση διακοπής παροχής ρεύματος ή σε reset.
III. Μνήμη FLASH: 32Kb όπου τα 2Kb είναι δεσμευμένα από το Firmware. Τα 30Kb χρησιμοποιούνται για την αποθήκευση των προγραμμάτων αφού πρώτα μεταγλωττιστούν από τον υπολογιστή. Η μνήμη Flash επίσης δεν έχει απώλεια των δεδομένων της.

2.3 **Θύρες εισόδου/εξόδου (input/output) της πλακέτας Arduino.**

Η πλακέτα Arduino Uno που θα χρησιμοποιηθεί για την παρούσα εργασία έχει 14 θύρες εισόδου/εξόδου όπου μπορούν να χρησιμοποιηθούν ως input ή output.[4]. Πιο αναλυτικά οι θύρες 3,5,6,9,10 και 11 έχουν την δυνατότητα να λειτουργούν σαν θύρες εξόδου PWM (PulseWidthModulation) . Οι θύρες 0 και 1 χρησιμοποιούνται για την λήψη RX και μετάδοση TX σειριακών δεδομένων. Επίσης η θύρα 0 έχει την δυνατότητα να συνδεθεί με μια δεύτερη πλακέτα Arduino.

Οι θύρες 2 και 3 μπορούν να λειτουργούν σαν εξωτερικά interrupt. Αυτό παρέχει την δυνατότητα σε μια αλλαγή τάσης να πραγματοποιείται μια διακοπή του προγράμματος και η εκτέλεση μιας συγκεκριμένης συνάρτησης.

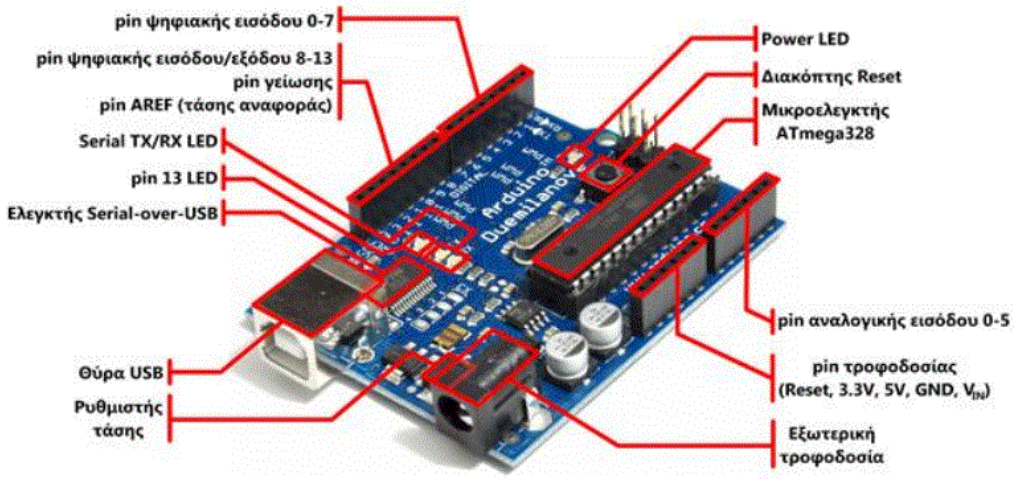
Τα pin όπου αναγράφεται από πάνω ANALOG IN με αρίθμηση από το 0 έως το 5 λειτουργούν ως αναλογική είσοδο με την χρήση του ADC (AnalogtoDigitalConverter) τα οποία έχουν την δυνατότητα μέσω του προγράμματος να χρησιμοποιηθούν σαν ψηφιακά pin εισόδου/εξόδου.

Στην συνέχεια υπάρχουν τα pin με την ένδειξη Power όπου κάθε pin έχει την εξής λειτουργία.

- **Reset**: όταν ενωθεί με την γείωση (GND) πραγματοποιεί μια επανεκκίνηση της πλακέτας.
- **3,3V**: Τροφοδοσία συσκευών ή αισθητήρων με τάση 3,3V όπου η τάση αυτή προέρχεται από την θύρα USB.
- **5V**: Τροφοδοσία συσκευών ή αισθητήρων με τάση 5V μέσω της θύρας USB είτε από την εξωτερική τροφοδοσία που μέσω του ρυθμιστή τάσης σταθεροποιείται στα 5V.
- **Pin 4,5 GND**:Γείωση.
- **Vin**: Χρησιμοποιείται είτε για την τροφοδοσία συσκευών με πλήρη τάση 7 έως 12V είτε με συνδυασμό της γείωσης (GND) να λειτουργήσει σαν εξωτερική τροφοδοσία του Arduino.

Η τροφοδοσία της συσκευής μπορεί να πραγματοποιηθεί με δύο τρόπους, είτε με την σύνδεση USB από έναν υπολογιστή ή μια εξωτερική τροφοδοσία μέσω του βύσματος που υπάρχει δίπλα από την θύρα Usb. Η θύρα USB χρησιμοποιείται και για την «φόρτωση»

(upload) του κώδικα (sketch). Η σειριακή επικοινωνία ανάμεσα στο μικροελεγκτή και τον υπολογιστή πραγματοποιείται μέσω του FDTI.

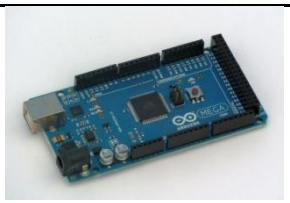


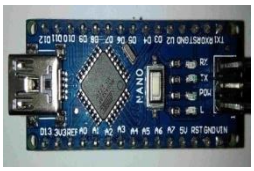
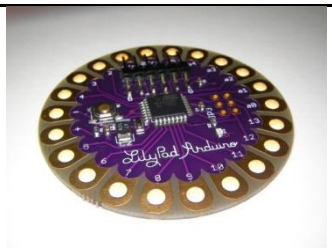
Εικόνα 1. Arduino Uno

2.4 Εκδόσεις Arduino

Στον παρακάτω πίνακα δίνονται οι εκδόσεις του Arduino. [5]

Έκδοση Arduino	Εικόνα πλακέτας
Arduino Uno Wifi	
Arduino /Genuino MKR1000	
Arduino 101, Genuino 101	

Έκδοση Arduino	Εικόνα πλακέτας
Arduino Zero	
Arduino Due	
Arduino Yun	
Arduino Leonardo	
Arduino Uno	
Arduino Mega2560	
Arduino Ethernet	
ArduinoFio	

Έκδοση Arduino	Εικόνα πλακέτας
Arduino Nano	
LilyPadArduino	
Arduino Pro	
Arduino Mega ADK	
ArduinoEsplora	
Arduino Micro	
Arduino Pro Mini	

Πίνακας 1 Εκδόσεις Arduino

2.5 **Arduino Shields**

Η πλακέτα arduino διαθέτει μια ευρεία γκάμα shields και modules . Τα κυριότερα εμφανίζονται στον παρακάτω πίνακα:

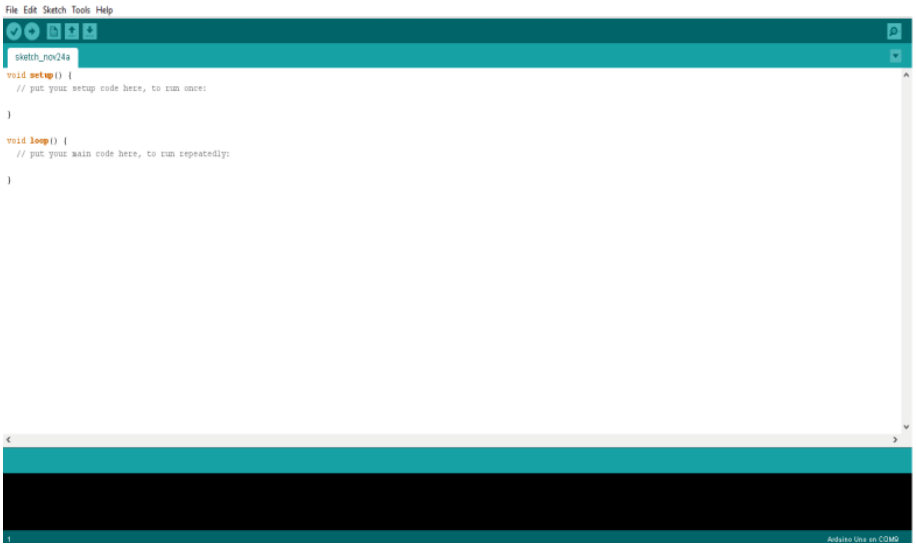
Ethernet Shield	Relay Shield	Proto Shield	Motor Shield	LCD Shield
Capacitive Shield	CAN-BUS Shield	Smoke detector Shield	NegativeVoltage Generation Shield	Wave Shield
CISECO ProtoX Shield	64-Button Shield	Joystik Shield	GSM/GPRS Shield	Gameduino Shield
MicroSD Shield	NFC/RFID Shield	AdafruitNeoPixel Shield	MP3 Player Shield	Camera Shield
GPS Logger Shield	Wireless SD Shield	Cc300 Wi-Fi Shield	ESP8266 Wi-Fi Shield	HC-05 Bluetooth Shield

Πίνακας 2. Arduino Shields

Εικόνα 2. Arduino Shields

2.6 Λογισμικό υλοποίησης της «έξυπνης μονάδας μετρήσεων».

Ξεκινώντας την υλοποίηση του project ξεκινάμε με το λογισμικό που θα χρησιμοποιηθεί για την συγγραφή του κώδικα. Για την πλακέτα Arduino υπάρχει το ArduinoIde. Το πρόγραμμα υπάρχει διαθέσιμο σε εκδόσεις για Mac, Linux και Windows.



Εικόνα 3. ArduinoIDE

Η δομή ενός project είναι πολύ απλή, υπάρχουν δυο συναρτήσεις οι οποίες είναι οι εξής:

- **Voidsetup()** : Στην συνάρτηση αυτή προσδιορίζουμε όλες τις εντολές που θέλουμε να εκτελεστούν μία και μόνο φορά. Κάθε φορά που αρχικοποιείται η πλακέτα (όταν συνδέεται με παροχή ρεύματος ή πατήσουμε το Reset) γίνεται η αρχικοποίηση των μεταβλητών και ο ορισμός των inputs/outputs που θα χρησιμοποιηθούν.
- **VoidLoop ()**: Στην συνάρτηση αυτή τοποθετείται όλος ο κώδικας, δηλαδή οι εντολές που είναι τοποθετημένες με την σειρά που έχουμε ορίσει θα εκτελεστούν. Εν συνεχεία μόλις φτάσει στο τέλος αυτόματα ενεργοποιείται η loop και οι εντολές ξεκινούν από την αρχή. Αυτό θα συμβαίνει όσο η πλακέτα θα είναι συνδεδεμένη με την παροχή ρεύματος.

3. Κατασκευή – Συνδεσμολογία

Η ιδέα για την δημιουργία του project προήλθε μετά από συζήτηση με έναν μελισσοκόμο σχετικά με τα προβλήματα που αντιμετωπίζει με τις αποστάσεις. Η βασική ιδέα ήταν να μπορεί να ενημερώνεται οποιαδήποτε στιγμή της ημέρας για την θερμοκρασία- υγρασία που επικρατεί στο σημείο που είναι τοποθετημένα τα μελίσσια αλλά και το βάρος των κυψελών χωρίς να είναι απαραίτητη η φυσική του παρουσία στον χώρο.

3.1 Arduino Uno

Ξεκινώντας για να μπορέσει να λειτουργήσει η μονάδα μετρήσεων που θα δημιουργήσουμε θα πρέπει να υπάρχει ο κατάλληλος εξοπλισμός. Οι βασικές πλακέτες που θα χρησιμοποιηθούν θα είναι οι εξής:

- Arduino Uno R3
- Arduino GSM Shield 2- Antenna Connector

Οι δύο πλακέτες συνδέονται η μία πάνω την άλλη έτσι ώστε να μπορούν να τροφοδοτούνται με ρεύμα. Η βάση είναι το ArduinoUnoR3 και από πάνω το GSMShield. Η πλακέτα του GSMShield δεν μπορεί να λειτουργήσει χωρίς την ύπαρξη του ArduinoUno όπως φαίνεται στην παρακάτω εικόνα.

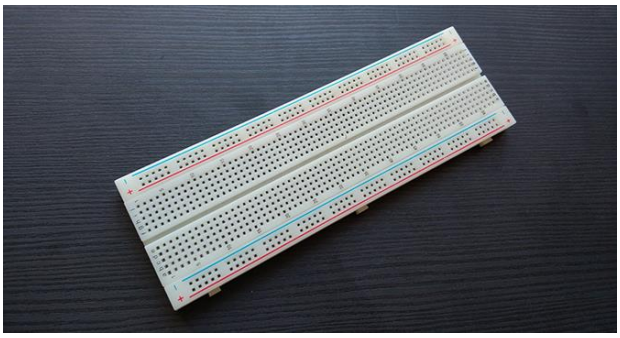
Εικόνα4. ArduinoUno&GSMShield

Ο λόγος που δεν μπορεί να λειτουργήσει μόνο του το gsmshiled είναι γιατι αρχικά δεν υπάρχει η δυνατότητα τροφοδοσίας με ρεύμα διότι δεν υπάρχει θύρα εισαγωγής καλωδίου ρεύματος και δεύτερον δεν υπάρχει τρόπος αποστολής των εντολών/δεδομένων στην πλακέτα.

Κάθε πλακέτα από τις δύο θα πρέπει να κωδικοποιηθεί διαφορετικά για να μπορέσουν να λειτουργήσουν ορθά.

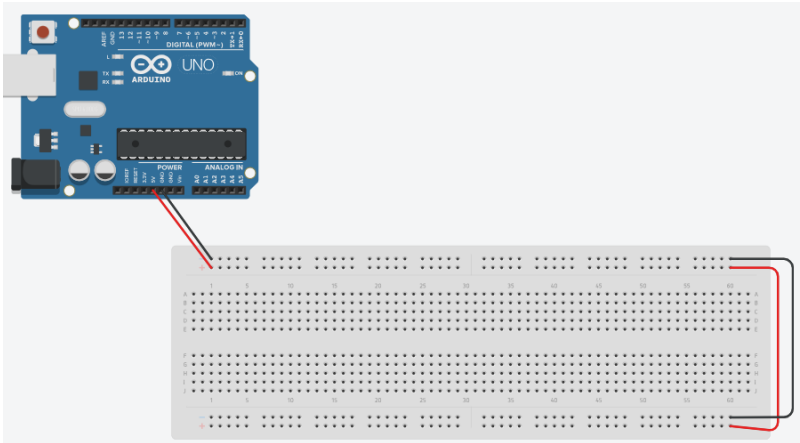
3.2 Breadboard

Στην συνέχεια θα πρέπει να υπάρχει ένα breadboard (πλακέτα διασύνδεσης χωρίς κολλήσεις) για να μπορέσουν εκεί να συνδεθούν και τα υπόλοιπα παρελκόμενα που θα χρησιμοποιηθούν για την δημιουργία της μονάδας.



Εικόνα 4. Breadboard (πλακέτα διασύνδεσης χωρίς κολλήσεις)

Η πλακέτα στα δύο άκρα της (πάνω και κάτω) έχει pin για την τροφοδοσία με ρεύμα αλλά και την γείωση που παρέχεται από το ArduinoUno τα οποία συνδέονται από την μία πλευρά και στην συνέχεια με jumpwire συνδέουν και την απέναντι πλευρά όπως φαίνεται και στην εικόνα παρακάτω.

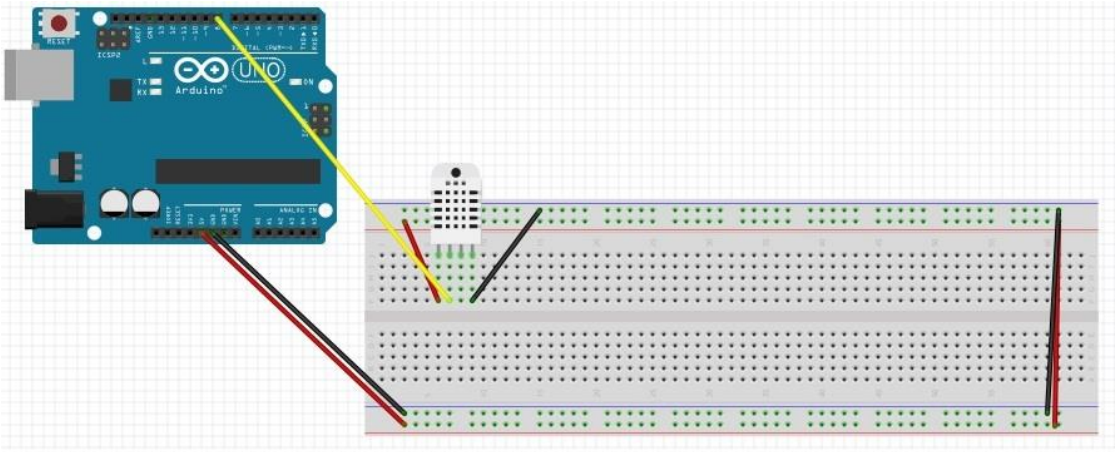


Εικόνα 5. Σύνδεση breadboar με ArduinoUno.

Η πλακέτα διαχωρίζεται στην μέση με το κενό που υπάρχει όπου η κάθε πλευρά τροφοδοτείται με ρεύμα από την αντίστοιχη πλευρά με τα pin. Κάθε μεριά θα πρέπει να συνδεθεί με τροφοδοσία για να μπορέσει να λειτουργήσει διαφορετικά οτιδήποτε συνδεθεί δεν θα είναι σε θέση να λειτουργήσει. Κάθε αντικείμενο που θα τοποθετηθεί πάνω στο breadboard θα πρέπει να συνδεθεί με ένα pin πάνω στην πλακέτα του Arduino για να μπορέσει να στείλει ή να λάβει δεδομένα.

3.3 **Αισθητήρας DHT11**

Αρχικά θα τοποθετήσουμε τον αισθητήρα DHT11 που είναι για να μπορέσουμε να μετρήσουμε την θερμοκρασία και την υγρασία του περιβάλλοντος. Το DHT11 διαθέτει 4 ‘‘ποδαράκια’’ σύνδεσης. Στο πρώτο συνδέεται η τροφοδοσία του , από το δεύτερο λαμβάνει και στέλνει τα δεδομένα , το τρίτο δεν είναι συνδεδεμένο και το τέταρτο συνδέεται με την γείωση.[6]



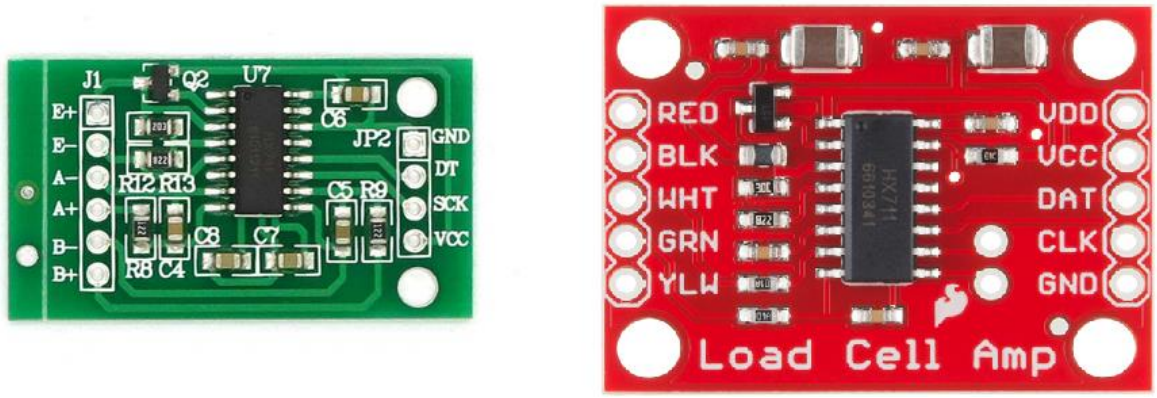
Εικόνα 6. Σύνδεση αισθητήρα DHT11.

Όπως φαίνεται και στην εικόνα 7 τοποθετούμε τον αισθητήρα DHT11 πάνω στο breadboardκαι στην συνέχεια κάνουμε τις συνδέσεις. Αρχικά με το κόκκινο καλώδιο από το πρώτο ποδαράκι τον συνδέουμε την παροχή ρεύματος. Με το κίτρινο καλώδιο το συνδέουμε με το Arduino στο pin 8 όπως έχει οριστεί και στον κώδικα και τέλος το γειώνουμε με το μαύρο καλώδιο στην αντίστοιχη θέση στο breadboard.

3.4 **Load Cell Amplifier**

Στην συνέχεια θα πρέπει να συνδέσουμε τον loadcell (αισθητήρας για την μέτρηση βάρους). Για να μπορέσει να συνδεθεί με το Arduino θα πρέπει ενδιάμεσα να υπάρχει και μια πλακέτα HX711 για την αποκωδικοποίηση των δεδομένων που στέλνει ο load cell με

κάθε αλλαγή της αντίστασης όπου με την κατάλληλη μετατροπή θα γίνεται η μέτρηση του βάρους.



Εικόνα8. HX711 LoadCellAmplifier

Η πλακέτα διαθέτει 5 συνδέσεις από την αριστερή μεριά και άλλες 5 από την δεξιά πλευρά. Στα αριστερά υπάρχουν οι συνδέσεις RED (E+) ,BLK (E-) ,WHT (A+) ,GRN (A-) ,YLW όπου τοποθετούνται τα καλώδια από τον loadcellανάλογα με το χρώμα τους στις αντίστοιχες θέσεις όπου REDείναι το κόκκινο καλώδιο που είναι του ρεύματος, BLKμαύρο καλώδιο που είναι για την γείωση , WHTλευκό καλώδιο για αποστολή-λήψη δεδομένων , GRNπράσινο καλώδιο για αποστολή-λήψη δεδομένων και YLWτο κίτρινο καλώδιο όπου δεν είναι γεφυρομένο με τον loadcellαλλά λειτουργεί ως εξωτερική γείωση εάν χρειαστεί. Σε μερικές περιπτώσεις το κίτρινο καλώδιο δεν υπάρχει σε κάποιους loadcell.

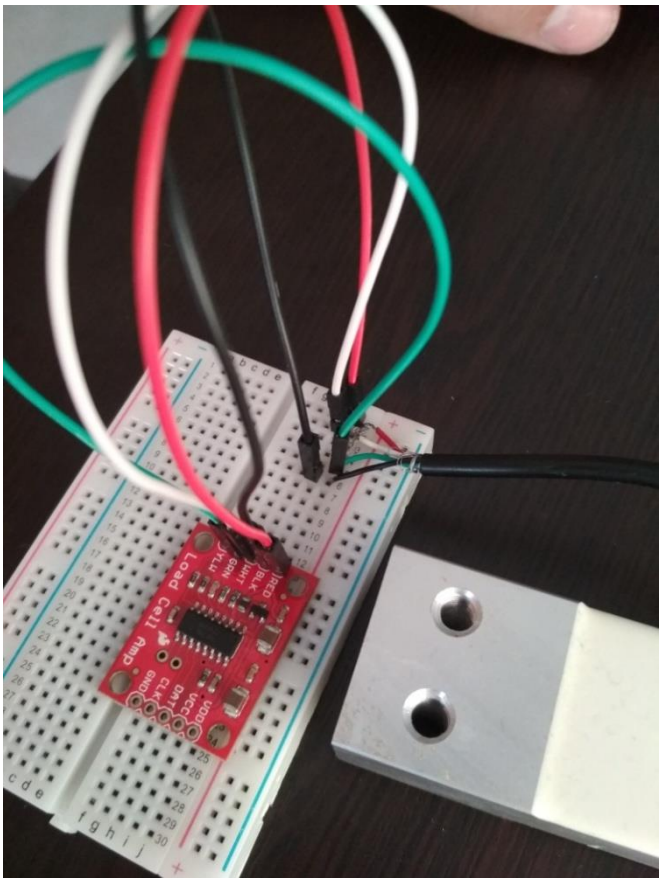
Στην δεξιά πλευρά υπάρχουν οι συνδέσεις VDD,VCC,DAT (SCK) ,CLK,GNDόπου κάθε μία έχει ξεχωριστή λειτουργία. Στην θέση VDDδεν συνδέεται τίποτα, στο VCC συνδέεται η παροχή ρεύματος 5V, οι θέσεις DATκαι CLK συνδέονται στα αντίστοιχα pin5,6 του Arduino οπου χρησιμοποιούνται ως DATA και CLOCKκαι τέλος η θέση GND συνδέεται η γείωση της πλακέτας.

3.5 **Load Cell**

Υπάρχουν πολλοί loadcellόπου χωρίζονται σε κατηγορίες ανάλογα με τα κιλά που μπορούν να ζυγίσουν. Διατίθενται από 100gέως και 200kg. Στην παρούσα εργασία θα χρησιμοποιηθεί ένας loadcell 50Kg.

Εικόνα 7.LoadCell 50Kg

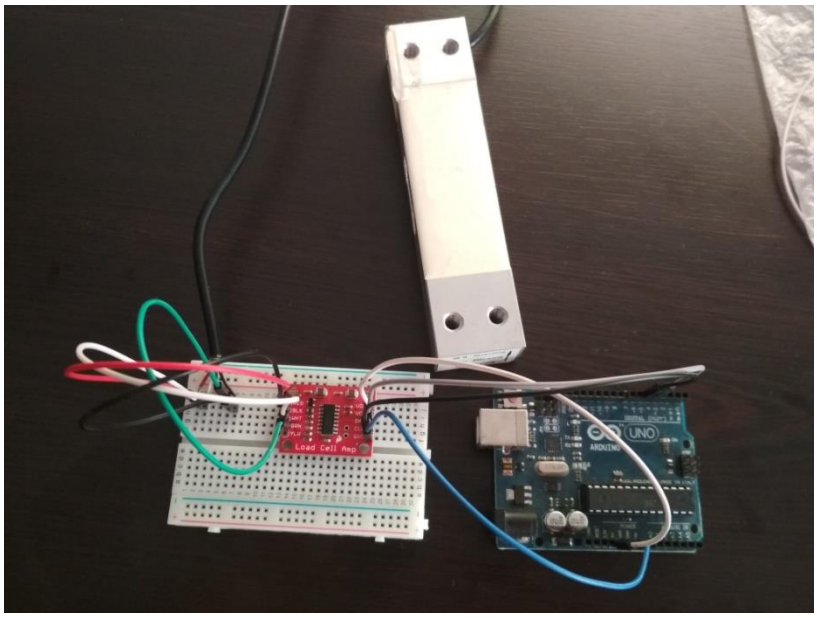
Αφού έχουμε όλα τα αντικείμενα που χρειαζόμαστε θα πρέπει να πραγματοποιήσουμε την σύνδεση όπως αναφέρθηκε προηγουμένως. Αρχικά θα πρέπει να συνδέσουμε τον loadcellμε το loadcellamplifier. Ξεκινώντας θα τοποθετήσουμε το κόκκινο καλώδιο στην θέση E+ για την τροφοδοσία , το μαύρο καλώδιο στο E- για την γείωση , το πράσινο καλώδιο στο A- και το λευκό καλώδιο στην θέση A+ για την αποστολή-λήψη δεδομένων .[7]



Εικόνα 8. Σύνδεση Loadcellμε HX711 amplifier

Αφού έχουμε πραγματοποιήσει τις συνδέσεις του load cell με το amplifier θα πρέπει να συνδέσουμε και το amplifierμε το Arduino. Αρχικά θα πρέπει να τροφοδοτήσουμε με ρεύμα την πλακέτα συνδέοντας το VCC με το pinτης τροφοδοσίας πάνω στο breadboard και αντίστοιχα να του τοποθετήσουμε και την γείωση του με το pin GND στο αντίστοιχο του breadboard. Έπειτα θα πρέπει να γεφυρώσουμε το amplifierμε το arduino για την επικοινωνία τους (αποστολή- λήψη δεδομένων) . Θα τοποθετήσουμε το pin DT στο pin4 του arduino και το SCKστο pin 5.

Μόλις έχουν ολοκληρωθεί όλες οι παραπάνω ενέργειες για να είμαστε σε θέση να πραγματοποιήσουμε ένα ακριβές ζύγισμα θα πρέπει να καλιμπράρουμε τον loadcell. Κάθε loadcell καλιμπράρετε με διαφορετικό αριθμό . Στον συγκεκριμένο που θα χρησιμοποιήσουμε καλιμπράρετε σωστά με την εντολή LoadCell.setCalFactor(696.0);



Εικόνα 9. Σύνδεση loadcellμε Arduino

3.6 **GSM Shield**

Εφόσον έχουμε δημιουργήσει την μονάδα μετρήσεων θα πρέπει να όλα τα δεδομένα που συλλέγονται (θερμοκρασία-υγρασία-βάρος) να αποθηκεύονται ή να υπάρχει η δυνατότητα ανά πάσα στιγμή να μπορούμε να ενημερωθούμε για το τι συμβαίνει. Γι’αυτό τον σκοπό θα χρησιμοποιήσουμε το gsmshield. Το gsmshield είναι ένα module που συνδέεται απευθείας πάνω στο arduino και δίνει την δυνατότητα σύνδεσης στο δίκτυο για αποστολή των δεδομένων σε έναν server ή την αποστολή μέσω sms. [8]

Η πλακέτα διαθέτει 3 διαφορετικές ενδείξεις φωτισμού:

- **Power:** Ενεργοποιείται όταν τροφοδοτείται με ρεύμα η πλακέτα.
- **Status:** Ενεργοποιείται μόλις ξεκινήσει η μετάδοση-λήψη δεδομένων με το gsm/gprs δίκτυο.
- **Net:** Ενεργοποιείται μόλις ξεκινήσει η διεπαφή με το δίκτυο.

Για την ορθή λειτουργία της πλακέτας συνίσταται μια εξωτερική πηγή τροφοδοσίας διότι με την απλή σύνδεση μέσω USB δεν παρέχεται το κατάλληλο ρεύμα που απαιτείται για να λειτουργήσει σωστά. Η προτεινόμενη τάση εισόδου είναι τα 5Vμε 1A. Με οποιοδήποτε λιγότερο πιθανόν η συσκευή να μην μπορεί να ανταποκριθεί στις εκάστοτε ανάγκες.

3.7 Προγραμματισμός της έξυπνης μονάδας

Στην συνέχεια θα αναλυθεί ο κώδικας του project. Πριν τις δύο αυτές συναρτήσεις γίνεται ο ορισμός όλων τον βιβλιοθηκών που θα χρησιμοποιηθούν στο πρόγραμμα μας.

#include <avr/wdt.h>	Παρέχει όλες τις προκαθορισμένες λειτουργίες του watchdogtimer.
#include <avr/interrupt.h>	Βιβλιοθήκη για τα την διαχείριση των interrupts.
#include <avr/sleep.h>	Βιβλιοθήκη για την λειτουργία sleep
#include <avr/power.h>	Βιβλιοθήκη για την διαχείριση της τροφοδοσίας.
#include <GSM.h>	Βιβλιοθήκη για τις λειτουργίες του GSMModule
#include <ArduinoHttpClient.h>	Βιβλιοθήκη για την διεπαφή του Arduino με WebServers.
#include <SimpleDHT.h>	Βιβλιοθήκη για τον αισθητήρα θερμοκρασίας-υγρασίας DHT.
#include <HX711_ADC.h>	Βιβλιοθήκη για την πλακέτα HX711.

Πίνακας 3 Βιβλιοθήκες

Στην συνέχεια μετά την αρχικοποίηση των βιβλιοθηκών θα πρέπει να γίνει ο ορισμός των μονάδων και θα γίνει δήλωση των pin εισόδου/εξόδου ανάλογα με τοντρόπο λειτουργίας του και ο ορισμός των δεδομένων για τα modules που θα χρησιμοποιηθούν.

const char PINNUMBER[ ] = "" const char GPRS_APN[ ] = "internet" const char GPRS_LOGIN[ ] = "" const char GPRS_PASSWORD[ ] = "" GSMClient client GPRS gprsAccess GSM gsmAccess volatileintnbr_remaining;	Σε αυτό το κομμάτι γίνεται ο ορισμός των δεδομένων για το GSMModule όπου δίνονται τα στοιχεία σχετικά με το Pin της κάρτας sim και όλα τα δεδομένα για την χρήση του GSM όπου χρειάζονται για κάθε πάροχο κινητής τηλεφωνίας. Στην συνέχεια γίνεται ο ορισμός μιας μεταβλητής όπου αρχικοποιείται ως μια μεταβλητή η οποία μπορεί να τροποποιείται από το πρόγραμμα βάση τις ανάγκες του προγράμματος για τις διακοπές (interrups) που θα πραγματοποιηθούν.

Πίνακας 4. Ορισμός δεδομένων GSM & Interrupts

Έπειτα θα πρέπει να οριστούν όλες οι παράμετροι όπου αφορούν τον APIServer που θα στέλνονται όλα τα δεδομένα που συλλέγονται από την «έξυπνη μονάδα μετρήσεων».

char server[ ] = "labmanager.teiep.gr" char path[ ] = "/SensorReading/write_data.php" int port = 80 int pinDHT11 = 8 intsleepStatus = 0 long t float load = 0 intstatusCode = 0 String response = " "	Αρχικά ορίζεται ο server όπου θα στέλνονται τα δεδομένα και στην συνέχεια δηλώνεται το αρχείο .php από το οποίο θα παίρνει τα δεδομένα. Επίσης δηλώνεται η «πόρτα» που θα χρησιμοποιηθεί για την σύνδεση και σε ποιο pin είναι συνδεδεμένος ο αισθητήρας DHT11. Έπειτα δηλώνονται και οι μεταβλητές για το

String okmsg = "ok" Stringerrormsg = "error"	counter του sleepstatus που θα αυξάνεται κάθε φορά που η πλακέτα θα μπαίνει σε sleepmode και η μεταβλητή t που θα μετράει τα milliseconds και τέλος κάποια strings που θα εμφανιστούν.

Πίνακας 5. Ορισμός server, DHT11, Counter

HX711_ADC LoadCell(4, 5) HttpClienthttpClient = HttpClient(client, server, port)	Στις δύο αυτές εντολές ορίζεται σε ποια pin (4 και 5) είναι συνδεδεμένος ο LoadCell (ζυγαριά) έτσι ώστε να μπορέσει να λάβει τα δεδομένα και ποια δεδομένα θα πάρει ο HttpClient όπου έχουν οριστεί νωρίτερα για να μπορέσει να λειτουργήσει σωστά.

Πίνακας 6. Ορισμός pin για Load cell & Δεδομένα του HttpClient

ISR(WDT_vect)
{
if (nbr_remaining> 0)
{
nbr_remaining = nbr_remaining - 1;
wdt_reset();
}
else
{
MCUSR = 0;
	Για την ορθή λειτουργία του WatchdogTimer υπάρχει η συνθήκη ISR(WDT_vect) όπου περιλαμβάνει μια if η οποία ελέγχει εάν η πλακέτα βρίσκεται σε sleepmode και μειώνει τα loops κατά 1 κάθε φορά διαφορετικά γίνεται ένα reboot και μηδενίζονται όλα τα flags.

Πίνακας 7 . Ορισμός συνθήκης ISR

WDTCSR |= 0b00011000;
WDTCSR = 0b00001000 | 0b000000;
while (1);
}
}
void configure_wdt(void)
	Σε αυτό το σημείο ορίζεται ο watchdogtimer να είναι σε κατάσταση reset όποτε χρειαστεί και στην συνέχεια ορίζεται να είναι σε κατάσταση για να πραγματοποιεί τα interrupts που χρειαζόμαστε ανάλογα

{
cli();
MCUSR = 0;
WDTCSR |= 0b00011000;
WDTCSR = 0b01000000 | 0b100001;
sei();
}
	με τα δευτερόλεπτα που του έχουμε ορίσει

Πίνακας 8. Ορισμός Whatchdogtimer σε reset mode και ορισμός δευτερολέπτων λειτουργίας

Τα δευτερόλεπτα που θα πραγματοποιεί τα interrupts ορίζονται ως εξής:

- 16 ms: 0b000000
- 500 ms: 0b000101
- 1 second: 0b000110
- 2 seconds: 0b000111
- 4 seconds: 0b100000
- 8 seconds: 0b100001

void sleep(intncycles)
{
nbr_remaining = ncycles;
set_sleep_mode(SLEEP_MODE_PWR_DOWN);
power_adc_disable();
while (nbr_remaining> 0)
{
sleep_mode();
sleep_disable();
}
power_all_enable();
}
	Στην συνέχεια ορίζεται η κλάση Sleep στην οποία μόλις μπαίνει σε sleepmode το Arduino θα εκτελεί τις παραπάνω εντολές. Αρχικά δίνεται για πόσους κύκλους θα είναι σε sleepmode και έπειτα μπαίνει σε κατάσταση sleep. Εκεί υπάρχει μια while στην οποία όσο ο αριθμός είναι μεγαλύτερη από το μηδέν συνεχίζει να βρίσκεται σε κατάσταση sleep.

Πίνακας 9. Sleep mode

Αφότου έχουν ολοκληρωθεί οι κύκλοι και μόνο με ένα interrupt που έχει οριστεί νωρίτερα μπορεί να βγει εκτός sleepmode απενεργοποιείται και στην συνέχεια ενεργοποιούνται όλα από την αρχή.

Έπειτα από όλες τις αρχικοποιήσεις θα δομηθούν οι δύο κύριες συναρτήσεις του project όπως αναφέραμε και προηγουμένως. Θα ξεκινήσουμε με την VoidSetup () και στην συνέχεια με την VoidLoop().

Void setup() {
Serial.begin(9600);
LoadCell.begin();
long stabilisingtime = 2000;
LoadCell.start(stabilisingtime);
LoadCell.setCalFactor(696.0);
configure_wdt();
}
	Σε αυτό το κομμάτι ορίζουμε όλες τις λειτουργίες του project που θέλουμε να ενεργοποιηθούν και να οριστούν. Αρχικά με την εντολή Serial.begin(9600); ορίζεται στα πόσα bits/δευτερόλεπτο θα ξεκινήσει η αναμετάδοση δεδομένων από τις πόρτες (pins). Έπειτα ενεργοποιείται ο LoadCell όπου με την εντολή ***LoadCell.setCalFactor(696.0);***καλιμπράρεται για την σωστή του λειτουργία και την ακρίβεια στα αποτελέσματα και αρχικοποιείται ο watchdogtimer με την εντολή ***configure_wdt();.***

Πίνακας 10. Void setup ( )

void loop() {
wdt_disable();
if ((gsmAccess.begin() != GSM_READY) &&
(gprsAccess.attachGPRS(GPRS_APN,
GPRS_LOGIN, GPRS_PASSWORD) !=
GPRS_READY)) {
gsmAccess.shutdown();
	Αρχικά απενεργοποιείται ο watchdogtimer ο οποίος θα ενεργοποιηθεί στην στην συνέχεια μόλις χρειαστεί. Στην συνέχεια υπάρχει μια συνθήκη if στην οποία ελέγχει εάν το GSM λειτουργεί σωστά και εάν τα δεδομένα είναι σωστά έτσι ώστε εάν κάποιο από τα στοιχεία του δεν είναι σωστό να απενεργοποιείται.

} else {
delay(1000);
byte temperature = 0;
byte humidity = 0;
interr = SimpleDHTErrSuccess;
	Στο δεύτερο σκέλος της if εφόσον η πρώτη συνθήκη δεν είναι true τότε εκτελούνται οι εντολές στο δεύτερο σκέλος (else) όπου ξεκινάει με μια καθυστέρηση και την αρχικοποίηση των μεταβλητών temperature και

	humidity στο μηδέν και τον ορισμό της μεταβλητής err για τον αισθητήρα DTH11 που θα χρησιμοποιηθεί στην συνέχεια για τυχών κάποιου σφάλματος.

Πίνακας 11. Void loop ( )

if ((err = dht11.read(&temperature, &humidity, NULL)) != SimpleDHTErrSuccess) {
delay(1000);
return;
}
	Στην επόμενη συνθήκη if πραγματοποιείται ο έλεγχος ότι ο αισθητήρας θερμοκρασίας – υγρασίας λειτουργεί κανονικά.

Πίνακας 12. Έλεγχος DHT11

t = millis();
while (millis() < t + 250) {
LoadCell.update();
}
load = LoadCell.getData();
intres_connect;
res_connect = httpClient.connect(server, port);
Serial.print(res_connect);
	Σε αυτό το κομμάτι ορίζεται η μεταβλητή t η οποία περιέχει κάποια millisecond όπου στην συνέχεια χρησιμοποιείται στην while η οποία περιέχει μια εντολή όπου συνεχώς ανανεώνει τον LoadCell. Στην συνέχεια τα δεδομένα του loadcell αποθηκεύονται στην μεταβλητή load. Τέλος ορίζεται η μεταβλητή res_connect όπου θα χρησιμοποιηθεί στην συνέχεια στην συνθήκη if.

Πίνακας 13. Ορισμός μεταβλητής t & res_connect

if (res_connect) {
httpClient.print("GET /SensorReading/write_data.php?");
httpClient.print("Temperature=");
httpClient.print((int)temperature);
httpClient.print("&Load=");
httpClient.print((int)abs(load));
httpClient.print("&Humidity=");
	Με την συνθήκη if(res_connect){ ελέγχεται εάν έχει πραγματοποιηθεί η σύνδεση με τον httpClient και μόλις αυτό είναι true τότε παίρνει όλα τα δεδομένα από το write_data.php και στην συνέχεια εμφανίζονται όλα τα δεδομένα από τους αισθητήρες που είναι

httpClient.print((int)humidity);
httpClient.println(" HTTP/1.1");
httpClient.print("Host: ");
httpClient.println(server);
httpClient.println("Connection: close");
httpClient.println();
	συνδεδεμένες με την πλακέτα.

Πίνακας 14. Έλεγχος σύνδεσης και εγγραφή δεδομένων

boolean test = true;
while (test) {
 	
if (httpClient.available()) {
char c = httpClient.read();
}
if (!httpClient.connected()) {
httpClient.stop();
test = false;
}
}
} else {
}
}
gsmAccess.shutdown();
delay(100);
configure_wdt();
delay(100);
sleep(20);
}
	Τέλος γίνονται οι έλεγχοι ότι ο httpclient είναι διαθέσιμος και διαβάζει τα δεδομένα και στην συνέχεια εάν δεν είναι συνδεδεμένος ο client τότε γίνεται τερματισμός και γίνεται απενεργοποίηση του gsm και του watchdogtimer και στην συνέχεια μπαίνει σε sleepmode.

Πίνακας 15. Έλεγχος HttpClient & απενεργοποίηση gsm

3.8 Λειτουργία συστήματος & Τρόπος χρήσης

3.8.1 Λειτουργία συστήματος

Μόλις έχει ολοκληρωθεί όλη η συνδεσμολογία της έξυπνης μονάδας και έχει πραγματοποιηθεί και ο προγραμματισμός αυτής ξεκινάει το κομμάτι που βλέπουμε πως λειτουργεί.

Αρχικά η έξυπνη μονάδα στο περισσότερο κομμάτι της ημέρας θα βρίσκεται σε κατάσταση sleep για εξοικονόμηση ενέργειας. Είναι προγραμματισμένη να ενεργοποιείται κάθε 8 ώρες μέσω ενός interrupt και με την βοήθεια του watchgodtimer. Κάθε φορά που θα ενεργοποιείται θα πραγματοποιεί τα εξής βήματα:

1. Πραγματοποίηση σύνδεσης του gsmshieldμε το gsmδίκτυο
2. Σύνδεση του Arduinoμε τον server
3. Αρχικοποίηση του αισθητήρα DHT11
4. Αρχικοποίηση του Whatchdogtimer
5. Ενεργοποίηση/Απενεργοποίηση του sleep mode
6. Αρχικοποίηση του load cell

Αφότου έχουν πραγματοποιηθεί τα παραπάνω χωρίς να υπάρχει κάποιο πρόβλημα ξεκινάει η διαδικασία εισόδου της πλακέτας στην λειτουργία επανεκτέλεσης εντολών (loop) όπου η μονάδα ξεκινάει και πραγματοποιεί τις μετρήσεις που απαιτούνται:

- Θερμοκρασία
- Υγρασία
- Βάρος

Στην συνέχεια εφόσον έχουν πραγματοποιηθεί όλες οι μετρήσεις τα αποτελέσματα των μετρήσεων έχουν αποθηκευτεί σε μια βάση δεδομένων με την βοήθεια του gsmshield αποστέλλονται στον server και εν συνεχεία εμφανίζονται στην ιστοσελίδα που έχουμε επιλέξει.

Τέλος μετά από την ολοκλήρωση του κύκλου εντολών η μονάδα εισέρχεται σε κατάσταση sleepmode μέχρις ότου να υπάρξει το επόμενο interruptπου έχει οριστεί για την πραγματοποίηση ενός νέου κύκλου εντολών όπως αναφέρθηκε προηγουμένως.

3.8.2 Τρόπος χρήσης έξυπνης μονάδας

Ο τρόπος χρήσης της μονάδας μπορεί να διαφέρει ανάλογα με τον εκάστοτε χρήστη. Στην παρούσα εργασία η μονάδα χρησιμοποιήθηκε για την παρακολούθηση μελισσών και τον περιβάλλοντα χώρο όπου βρίσκονται τα μελίσσια. Ο σκοπός της παρακολούθησης ήταν να μπορεί ο μελισσοκόμος να γνωρίζει τις καιρικές συνθήκες που επικρατούν καθώς και το φορτίο της κηρήθρας κάθε μέρα ανά 8 ώρες που πραγματοποιούνται οι μετρήσεις.

Ειδικότερα τις ημέρες όπου πραγματοποιείται παραγωγή μελιού υπάρχει και η δυνατότητα παρακολούθησης της παραγωγής που πραγματοποιείται ανά ώρα,ημέρα, εβδομάδα , μήνα. Αυτό έχει ως σκοπό να μπορέσει να αντλήσει κάποια συμπεράσματα για κάθε μελισσοκομική περίοδο και να πραγματοποιεί μια σύγκριση των αποτελεσμάτων συγκρίνοντας τις καιρικές συνθήκες έτσι ώστε ο παραγωγός να γνωρίζει ποιες είναι οι πιο κατάλληλες συνθήκες για την μεγαλύτερη παραγωγή.

3.9 **Δεδομένα αισθητήρων**

Για την παρουσίαση των αποτελεσμάτων των μετρήσεων υπάρχει μια σελίδα παρουσίασης δεδομένων. Για την αποστολή-λήψη δεδομένων αλλά και μορφοποίησης της σελίδας και τον τρόπο τον οποίο παρουσιάζονται τα αποτελέσματα θα χρησιμοποιηθούν 2 (δύο) αρχεία php.

3.9.1 Sensor Reading.php

Ξεκινώντας θα πρέπει να δημιουργηθεί ένα αρχείο php το οποίο θα λαμβάνει συνεχώς τα δεδομένα από το arduino και θα τα τοποθετεί στην βάση δεδομένων όπου αυτόματα μέσω του writedata.php θα εμφανίζονται απευθείας στην σελίδα προβολής δεδομένων. Στο αρχείο αυτό υπάρχουν όλες οι διαμορφώσεις που είναι απαραίτητες για τον σωστό διαχωρισμό των δεδομένων.

<?php $url=$_SERVER['REQUEST_URI']; header("Refresh: 180; URL=$url"); $dbusername = "arduino"; $dbpassword = "hasourakis"; $server = "localhost"; $dbconnect = mysql_pconnect($server, $dbusername, $dbpassword);	Στο πρώτο κομμάτι δίνονται οι παράμετροι έτσι ώστε να συνδεθεί ο server στην βάση δεδομένων και την εντολή να πραγματοποιεί ανανέωση της ιστοσελίδας ανά πέντε δευτερόλεπτα.

$dbselect = mysql_select_db("arduino",$dbconnect); ?> <?php	

Πίνακας 16. Παράμετροι ιστοσελίδας

$current_d = date('y.m.d.H');
for($hor=23;$hor>=0;$hor--)
{
$sel_query = "select DATE_FORMAT((select DATE_SUB('".$current_d."', INTERVAL $hor HOUR)),'%Y, %m, %d, %H') as hours";
$rmon = mysql_query($sel_query);
while($res_hours=mysql_fetch_assoc($rmon)){
$hours[] = $res_hours['hours'];
$disp_hours = explode(', ',$res_hours['hours']);
$disp_year[] = $disp_hours[0];
$disp_month[] = $disp_hours[1];
$disp_day[] = $disp_hours[2];
$disp_hour[] = $disp_hours[3];
	Σε αυτό το κομμάτι ορίζεται ο τρόπος που θα εμφανίζεται το timestamp δηλαδή το χρονικό σημείο που θα λαμβάνει τα δεδομένα από τους αισθητήρες ο server.

Πίνακας 17. Timestamp

for($hor=23;$hor>=0;$hor--)
{
$get_all_avgs = "select AVG(`Load`), AVG(`Humidity`), AVG(`Temperature`) from `sensor` WHERE ((extract(YEAR_MONTH from time) = '".$disp_year[$hor].$disp_month[$hor]."') & (extract(DAY_HOUR from time) = '".$disp_day[$hor].$disp_hour[$hor]."'))";
$result_avgs = mysql_query($get_all_avgs);
while($result_get_all_avgs = mysql_fetch_assoc($result_avgs))
{
$dayhourload[] = number_format((float)$result_get_all_avgs['AVG(`Load`)'
	Εν συνεχεία ορίζεται ο τρόπος που θα δημιουργού νται οι μέσοι όροι ανά κατηγορία (θερμοκρασία-υγρασία-βάρος).

```
], 2, '.', '');
$dayhourhum[] = number_format((float)$result_get_all_avgs['AVG(`Humidity`)'], 2, '.', '');
$dayhourTemp[] = number_format((float)$result_get_all_avgs['AVG(`Temperature`)'], 2, '.', '');
        }
}
```

Πίνακας 18. Μέσοι όροι θερμοκρασίας-υγρασίας-βάρους

```
$dayhourTable = array(
'cols' => array(
array('type' => 'number', 'label' => 'Load'),
 array('type' => 'number', 'label' => 'Humidity'),
array('type' => 'number', 'label' => 'Temperature')
        )
);
for($i=23;$i>=0;$i--)
{
$dayhourTable['rows'][] = array(
'c' => array (
array('v' => 'new Date('.$disp_year[$i].', '.($disp_month[$i] - 1).', '.$disp_day[$i].', '.$disp_hour[$i].', 0, 0, 0)'),
array('v' => $dayhourload[$i]),
array('v' => $dayhourhum[$i]),
array('v' => $dayhourTemp[$i])
   )
);
}
$on = str_replace('"new', "new", json_encode($dayhourTable));
$on_ = str_replace(')"', ")", $on);
```

Σε αυτό το κομμάτι παίρνει όλες τις μετρήσεις που δημιουργούνται ανά ημέρα και τις τοποθετεί σε έναν πίνακα ανά κατηγορία και ανά 8 ώρες που πραγματοποιούνται οι μετρήσεις.

```
?>
<?php
$date_we = date('Y-m-d');
$get_week_days = "select
DAYNAME(date_sub('$date_we', interval 6 day)) as
DAY_START,
date_sub('$date_we', interval 6 day) As sel_date_start,
DAYNAME(date_sub('$date_we', interval 5 day)) as
DAY_2,
date_sub('$date_we', interval 5 day) As sel_date_start2,
DAYNAME(date_sub('$date_we', interval 4 day)) as
DAY_3,
date_sub('$date_we', interval 4 day) As sel_date_start3,
DAYNAME(date_sub('$date_we', interval 3 day)) as
DAY_4,
date_sub('$date_we', interval 3 day) As sel_date_start4,
DAYNAME(date_sub('$date_we', interval 2 day)) as
DAY_5,
date_sub('$date_we', interval 2 day) As sel_date_start5,
DAYNAME(date_sub('$date_we', interval 1 day)) as
DAY_6,
date_sub('$date_we', interval 1 day) As sel_date_start6,
DAYNAME('$date_we') as DAY_END,
date_sub('$date_we', interval 0 day) As sel_date_end";
$result_week_days = mysql_query($get_week_days);
while($res_get_week_days =
mysql_fetch_assoc($result_week_days))
{
$dataFeildsArray[] =
$res_get_week_days['DAY_START']." - " .
$res_get_week_days['sel_date_start'];
   $dataFeildsArray[] = $res_get_week_days['DAY_2']." -
" . $res_get_week_days['sel_date_start2'];
```

```
    $dataFeildsArray[] = $res_get_week_days['DAY_3']." -
" . $res_get_week_days['sel_date_start3'];
    $dataFeildsArray[] = $res_get_week_days['DAY_4']." -
" . $res_get_week_days['sel_date_start4'];
    $dataFeildsArray[] = $res_get_week_days['DAY_5']." -
" . $res_get_week_days['sel_date_start5'];
    $dataFeildsArray[] = $res_get_week_days['DAY_6']." -
" . $res_get_week_days['sel_date_start6'];
    $dataFeildsArray[] =
$res_get_week_days['DAY_END']." - " .
$res_get_week_days['sel_date_end'];
}
```

Πίνακας 19. Δημιουργία πίνακα αποτελεσμάτων ανά 8 ώρες

```
for($i=6;$i>=0;$i--)
{
$get_all_avgs = "select AVG(`Load`), AVG(`Humidity`),
AVG(`Temperature`) from `sensor` WHERE
DATE_FORMAT(time, '%W - %Y-%m-%d') =
'".$dataFeildsArray[$i]."'";
$weekday = explode('-', $dataFeildsArray[$i]);
$weekday_month[] = $weekday[2];
$weekday_day[] = $weekday[3];
$result_avgs = mysql_query($get_all_avgs);
while($result_get_all_avgs =
mysql_fetch_assoc($result_avgs))
{
$weekdayload[] =
number_format((float)$result_get_all_avgs['AVG(`Load`)'
], 2, '.', '');
$weekdayhum[] =
number_format((float)$result_get_all_avgs['AVG(`Humidi
ty`)'], 2, '.', '');
$weekdayTemp[] =
```

Στην συνέχεια δημιουργείτε ένας πίνακας ο οποίος θα περιέχει όλες τι μετρήσεις ανά ημέρα της εβδομάδας για κάθε κατηγορία όπως φαίνεται στο παραπάνω κομμάτι κώδικα.

```
number_format((float)$result_get_all_avgs['AVG(`Tempe
rature`)'], 2, '.', '');
            }
}
$WeekdayTable = array(
'cols' => array(
array('type' => 'number', 'label' => 'Load'),
array('type' => 'number', 'label' => 'Humidity'),
array('type' => 'number', 'label' => 'Temperature')
    )
            );
for($i=6;$i>=0;$i--)
{
$WeekdayTable['rows'][] = array(
 'c' => array (
array('v' => 'new Date('.date('Y').', '.($weekday_month[$i]
- 1).', '.$weekday_day[$i].')'),
  array('v' => $weekdayload[$i]),
   array('v' => $weekdayhum[$i]),
array('v' => $weekdayTemp[$i])
            )
);
}
$WeekdayTableon = str_replace("'new', "new",
json_encode($WeekdayTable));
$WeekdayTableon_ = str_replace(')"', ")",
$WeekdayTableon);
?>
```

Πίνακας 20. Δημιουργία πίνακα αποτελεσμάτων ανά ημέρα

```
<?php
$month = date('m');
$year = date('Y');
for($m=1;$m<=cal_days_in_month(CAL_GREGORIAN,
```

Το επόμενο βήμα που πραγματοποιείται είναι η δημιουργία ενός πίνακα ο οποίος θα απεικονίζει τα

```
$month, $year);$m++)
{
if($m <10){ $mon="0".$m; }
else { $mon = $m;}
  $mon_days[] = $year.'-'.$month.'-'.$mon;
}
for($m=1;$m<=cal_days_in_month(CAL_GREGORIAN,
$month, $year);$m++)
{
$get_all_avgs = "select AVG(`Load`), AVG(`Humidity`),
AVG(`Temperature`) from `sensor` WHERE
DATE_FORMAT(time, '%Y-%m-%d') =
'".$mon_days[$m - 1]."'";

$result_avgs = mysql_query($get_all_avgs);
while($result_get_all_avgs =
mysql_fetch_assoc($result_avgs))
{
$MonthDayload[] =
number_format((float)$result_get_all_avgs['AVG(`Load`)'
], 2, '.', '');
$MonthDayhum[] =
number_format((float)$result_get_all_avgs['AVG(`Humidi
ty`)'], 2, '.', '');
$MonthDayTemp[] =
number_format((float)$result_get_all_avgs['AVG(`Tempe
rature`)'], 2, '.', '');
          }
}
$MonthDayTable = array(
'cols' => array(
array('type' => 'number', 'label' => 'Load'),
array('type' => 'number', 'label' => 'Humidity'),
```

αποτελέσματα των μετρήσεων ανά εβδομάδα μέσα σε κάθε μήνα την δεδομένη στιγμή που ο χρήστης εισέρχεται στην σελίδα.

```
array('type' => 'number', 'label' => 'Temperature')
    )
);
for($m=1;$m<=cal_days_in_month(CAL_GREGORIAN,
$month, $year);$m++)
{
$MonthDayTable['rows'][] = array(
'c' => array (
array('v' => 'new Date('.$year.', '.($month - 1).', '.$m.')'),
array('v' => $MonthDayload[$m-1]),
 array('v' => $MonthDayhum[$m-1]),
array('v' => $MonthDayTemp[$m-1])
)
);
}
$MonthDayTableon = str_replace('"new', "new",
json_encode($MonthDayTable));
$MonthDayTableon_ = str_replace(')"', ")",
$MonthDayTableon);
?>
```

Πίνακας 21. Δημιουργία πίνακα αποτελεσμάτων ανά εβδομάδα

```
<?php
for($i=52;$i>=0;$i--)
{
$WeeksYear[$i] =  date('Y-m-d', strtotime('-'.$i.'
week',strtotime($year.'-12-31') ));
$WeekYear = explode('-', $WeeksYear[$i]);
$WeekYear_year[] = $WeekYear[0];
$WeekYear_month[] = $WeekYear[1];
$WeekYear_day[] = $WeekYear[2];
}
for($i=52;$i>0;$i--)
{
```

Στο κομμάτι αυτό λαμβάνοντας ως δεδομένα όλους του μέσους όρους που υπάρχουν από τα προηγούμενα κομμάτια κώδικα δημιουργείτε ένας πίνακας ο οποίος απεικονίζει τα δεδομένα ανά 4 μήνες στον χρόνο και αντίστοιχα ανά 3 μήνες τον χρόνο.

```
$get_all_avgs = "select AVG(`Load`), AVG(`Humidity`), AVG(`Temperature`) from `sensor` WHERE DATE_FORMAT(time, '%Y-%m-%d') BETWEEN '".$WeeksYear[$i]."' AND '".$WeeksYear[$i-1]."'";
$result_avgs = mysql_query($get_all_avgs);
while($result_get_all_avgs = mysql_fetch_assoc($result_avgs))
{
$WeekYearload[] = number_format((float)$result_get_all_avgs['AVG(`Load`)'], 2, '.', '');
$WeekYearhum[] = number_format((float)$result_get_all_avgs['AVG(`Humidity`)'], 2, '.', '');
$WeekYearTemp[] = number_format((float)$result_get_all_avgs['AVG(`Temperature`)'], 2, '.', '');
        }
}
$WeekYearTable = array(
 'cols' => array( array('type' => 'date', 'label' => 'Week of Year'),
array('type' => 'number', 'label' => 'Load'),
array('type' => 'number', 'label' => 'Humidity'),
array('type' => 'number', 'label' => 'Temperature')
   )
);
for($i=52;$i>=0;$i--)
{
$WeekYearTable['rows'][] = array(
'c' => array (
array('v' => 'new Date('.$WeekYear_year[$i].', '.($WeekYear_month[$i] - 1).', '.$WeekYear_day[$i].')'),
```

```
array('v' => $WeekYearload[$i]),
array('v' => $WeekYearhum[$i]),
array('v' => $WeekYearTemp[$i])
        )
  );
}
$WeekYearTableon = str_replace("'new', "new",
json_encode($WeekYearTable));
$WeekYearTableon_ = str_replace(')"', ")",
$WeekYearTableon);
?>
```

Πίνακας 22 Δημιουργία πίνακα αποτελεσμάτων ανά μέσο όρο για 3 και 4 μήνες

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html lang="en">
<head>
<title>Load, Humidity, Temperature Remote
Service</title>
<meta http-equiv="Content-type"
content="text/html;charset=UTF-8">
<meta http-equiv="Content-Style-Type"
content="text/css">
<meta name="viewport" content="width=device-width,
initial-scale=1">
<link rel="stylesheet" type="text/css" media="all"
href="css/style.css">
```

Στην συνέχεια υπάρχει το κομμάτι της διαμόρφωσης της σελίδας. Ξεκινώντας ορίζεται ο τίτλος που φαίνεται σε κάθε καρτέλα στον περιηγητή και στην συνέχεια ορίζεται το link για να φορτωθούν οι προδιαγραφές σχεδίασης της σελίδας μέσω του css αρχείου.

Πίνακας 23. Διαμόρφωση σελίδας απεικόνισης αποτελεσμάτων

```
link
href='https://fonts.googleapis.com/css?family=Open+Sans:
300,400,600,700' rel='stylesheet' type='text/css'>
<link rel="stylesheet" type="text/css" href="Font-
Awesome-4.7/css/font-awesome.min.css">
```

Σε αυτό το κομμάτι κώδικα μέσω του GoogleChars δημιουργούνται όλοι οι πίνακες που απεικονίζονται στην σελίδα προβολής των

```
<link rel="stylesheet" type="text/css" href="css/bootstrap.min.css">
<link rel="stylesheet" type="text/css" href="css/templatemo-style.css">
<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
<script type="text/javascript" src="//ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js"></script>
<script type="text/javascript">
$(document).ready(function(){
    //$('#data').after('<div id="nav"></div>');
var g = document.createElement("div");
    g.id = 'nav';
g.className = 'links';
varpf = document.getElementById('pagefoot');
pf.appendChild(g);
varrowsShown = 34;
varrowsTotal = $('#data tbodytr').length;
varnumPages = rowsTotal/rowsShown;
    for(i = 0;i <numPages;i++) {
varpageNum = i + 1;
        $('#nav').append('<a href="#" rel="'+i+'">'+pageNum+'</a> ');
    }
    $('#data tbodytr').hide();
    $('#data tbodytr').slice(0, rowsShown).show();
    $('#nav a:first').addClass('active');
    $('#nav a').bind('click', function(){
        $('#nav a').removeClass('active');
        $(this).addClass('active');
varcurrPage = $(this).attr('rel');
varstartItem = currPage * rowsShown;
```

δεδομένων. Ορίζονται όλες οι τιμές που περιέχει κάθε πίνακας αλλά και με τον τρόπο τον οποίο θα προβάλλονται.

```
varendItem = startItem + rowsShown;
        $('#data
tbodytr').css('opacity','0.0').hide().slice(startItem, endItem).
css('display','table-row').animate({opacity:1}, 300);
    });
});
</script>
<script type="text/javascript">
google.charts.load('current', {packages: ['corechart', 'bar']});
google.charts.setOnLoadCallback(drawTrendlines);
function drawTrendlines() {
vardayhourdata = new
google.visualization.DataTable(<?php echo $on_; ?>);
varweekdaydata = new
google.visualization.DataTable(<?php echo
$WeekdayTableon_; ?>);
varMonthDaydata = new
google.visualization.DataTable(<?php echo
$MonthDayTableon_; ?>);
varWeekYeardata = new
google.visualization.DataTable(<?php echo
$WeekYearTableon_; ?>);
varMonthYeardata = new
google.visualization.DataTable(<?php echo
$MonthYearTableon_; ?>);
vardayhouroptions = {
title: 'Load, Humidity and Temperature Level Throughout
the Day',
trendlines: {
0: {type: 'linear', lineWidth: 3, opacity: .3},
          1: {type: 'linear', lineWidth: 6, opacity: .3},
          2: {type: 'linear', lineWidth: 9, opacity: .3}
        },
```

```
hAxis: {
title: 'Hours Before Now',
format: 'H:mm - E',
timeZone: '+2'
},
vAxis: {
title: 'Rating'
}
};
varweekdayoptions = {
title: 'Load, Humidity and Temperature Level Throughout the Week',
trendlines: {
0: {type: 'linear', lineWidth: 3, opacity: .3},
1: {type: 'linear', lineWidth: 6, opacity: .3},
2: {type: 'linear', lineWidth: 9, opacity: .3}
},
hAxis: {
        title: 'Day of Week',
        format: 'E'
        //viewWindow: {
         // min: [7, 30, 0],
          //max: [17, 30, 0]
       // }
       },
vAxis: {
        title: 'Rating'
       }
     };
varMonthDayoptions = {
title: 'Load, Humidity and Temperature Level Throughout the Month',
trendlines: {
```

```
          0: {type: 'linear', lineWidth: 3, opacity: .3},
        1: {type: 'linear', lineWidth: 6, opacity: .3},
        2: {type: 'linear', lineWidth: 9, opacity: .3}
      },
hAxis: {
        title: 'Day of Month',
        format: 'E - d'
        //viewWindow: {
         // min: [7, 30, 0],
         // max: [17, 30, 0]
        ///}
      },
vAxis: {
        title: 'Rating'
      }
    };
          varWeekYearoptions = {
     title: 'Load, Humidity and Temperature Level
Throughout the Year',
trendlines: {
        0: {type: 'linear', lineWidth: 3, opacity: .3},
        1: {type: 'linear', lineWidth: 6, opacity: .3},
        2: {type: 'linear', lineWidth: 9, opacity: .3}
      },
hAxis: {
        title: 'Week of Year',
        format: 'E MMM yyyy'
        //viewWindow: {
        //  min: [7, 30, 0],
        //  max: [17, 30, 0]
       // }
      },
vAxis: {
        title: 'Rating'
      }
```

```
    };
        varMonthYearoptions = {
      title: 'Load, Humidity and Temperature Level
Throughout the Year',
trendlines: {
        0: {type: 'linear', lineWidth: 3, opacity: .3},
        1: {type: 'linear', lineWidth: 6, opacity: .3},
        2: {type: 'linear', lineWidth: 9, opacity: .3}
      },
hAxis: {
        title: 'Month of Year',
        format: 'MMM yyyy'
        //viewWindow: {
        //  min: [7, 30, 0],
        //  max: [17, 30, 0]
        //}
},
vAxis: {
        title: 'Rating'
      }
    };
vardayhourchart = new
google.visualization.ColumnChart(document.getElementBy
Id('day_chart_div'));
dayhourchart.draw(dayhourdata, dayhouroptions);
        varweekdaychart = new
google.visualization.ColumnChart(document.getElementBy
Id('week_chart_div'));
weekdaychart.draw(weekdaydata, weekdayoptions);
varMonthDaychart = new
google.visualization.ColumnChart(document.getElementBy
Id('month_chart_div'));
MonthDaychart.draw(MonthDaydata, MonthDayoptions);
```

```
varWeekYearchart = new
google.visualization.ColumnChart(document.getElementBy
Id('week_year_chart_div'));
WeekYearchart.draw(WeekYeardata, WeekYearoptions);
varMonthYearchart = new
google.visualization.ColumnChart(document.getElementBy
Id('month_year_chart_div'));
MonthYearchart.draw(MonthYeardata,
MonthYearoptions);
}
</script>
</head>
<body>
<div class="container">
<section class="col-md-12 content" id="home">
<div class="col-lg-6 col-md-6 content-item tm-black-
translucent-bg tm-logo-box">
<i class="fafa-snowflake-o fa-4x tm-logo"></i>
<h1 class="text-uppercase">Load, Humidity,
Temperature</h1>
        <table id="data" class="redTable">
<thead>
<tr>
<th>ID</th>
                <th>Timestamp</th>
<th>Load</th>
        <th>Temperature</th>
        <th>Humidity</th>
        </tr>
</thead>
<tfoot>
<tr>
<td id="pagefoot" colspan="5">
```

```
</td>
</tr>
</tfoot>
<tbody>
<?php
```

Πίνακας 24. Δημιουργία γραφημάτων μέσω Google charts

```
$con=mysqli_connect("localhost","arduino","hasourakis","arduino");
$result = mysqli_query($con,'SELECT * FROM sensor ORDER BY id DESC');
$response = array();
$posts = array();
while($row = mysqli_fetch_array($result))
{
        $id=$row['id'];
        $time=$row['time'];
        $load=$row['Load'];
        $temp=$row['Temperature'];
        $hum=$row['Humidity'];
        echo "<tr>";
        echo "<td>" . $id . "</td>";
        echo "<td>" . $time . "</td>";
        echo "<td>" . $load . "</td>";
        echo "<td>" . $temp . "</td>";
        echo "<td>" . $hum . "</td>";
        echo "</tr>";
        $posts[id] = $id;
        $posts[time] = $time;
        $posts[load] = $value;
        $posts[temp] = $temp;
        $posts[hum] = $hum;
        $response[] = $posts;
}
```

Τέλος πραγματοποιείται σύνδεση με την βάση δεδομένων και γίνεται ανάκτηση των δεδομένων που είναι αποθηκευμένα και παρουσιάζονται και έπειτα η αποσύνδεση με την βάση.

```
mysqli_close($con);
?>
</tbody>
</table>
```

Πίνακας 25. Σύνδεση με βάση δεδομένων και ανάκτηση δεδομένων για παρουσίαση

3.9.2 Write data.php

Το αρχείο writedata.php είναι ο βασικός πυλώνας για τον τρόπο σύνδεσης αλλά και μετάδοσης των δεδομένων από το arduino στην σελίδα μας. Στην συνέχεια θα αναλυθεί ο κώδικας του αρχείου.

```
<?php
    $dbusername = "arduino";
    $dbpassword = "hasourakis";
    $server = "localhost";
```

Στο πρώτο κομμάτι του κώδικα δίνονται τα στοιχεία επαλήθευσης που χρειάζονται για να πραγματοποιηθεί η σύνδεση στην βάση δεδομένων.

Πίνακας 26. Ορισμός στοιχείων για σύνδεση με την βάση δεδομένων

```
$dbconnect = mysql_pconnect($server, $dbusername, $dbpassword)
$dbselect = mysql_select_db("arduino",$dbconnect)
```

Στην συνέχεια πραγματοποιείται η σύνδεση με την βάση.

Πίνακας 27. Σύνδεση με την βάση δεδομένων

```
$sql = "INSERT INTO arduino.sensor (`Load`, `Temperature`, `Humidity`) VALUES ('".$_REQUEST["Load"]."', '".$_REQUEST["Temperature"]."', '".$_REQUEST["Humidity"]."')"
mysql_query($sql)
```

Τέλος δίνονται οι βασικές εντολές για την sql έτσι ώστε να μπορέσει να λάβει τα δεδομένα από τους αισθητήρες βάρους – θερμοκρασίας – υγρασίας.

Πίνακας 28. Ορισμός εντολών για την λήψη δεδομένων από τους αισθητήρες

4. Αποτελέσματα & Προβλήματα

Σε αυτό το κεφάλαιο θα γίνει αναφορά στον τρόπο παρουσίασης των αποτελεσμάτων της πτυχιακής αλλά και τα προβλήματα που αντιμετωπίστηκαν κατά την διάρκεια εκπόνησης της.

4.1 Απεικόνιση αποτελεσμάτων

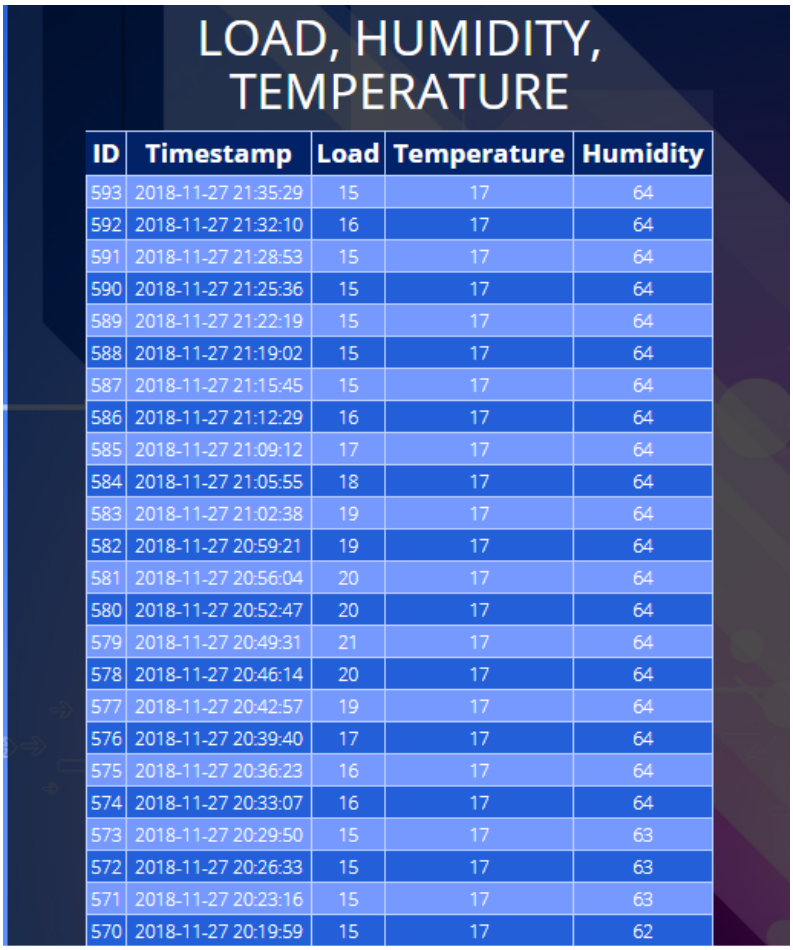
Η απεικόνιση των αποτελεσμάτων θα πραγματοποιείται μέσω της ιστοσελίδας https://sonetto.teiep.gr/SensorReading/ όπου θα μπορεί να εισέρχεται οποιαδήποτε στιγμή ο χρήστης.

Εικόνα 10. Σελίδα απεικόνισης αποτελεσμάτων

Όπως φαίνεται και στην εικόνα παραπάνω η σελίδα περιέχει δύο τρόπους απεικόνισης αποτελεσμάτων. Στα αριστερά εμφανίζονται όλες οι καταγραφές που πραγματοποιούνται ενώ στα δεξιά υπάρχουν όλοι οι πίνακες που δημιουργήθηκαν μέσω του phpκαι των googlechartsόπως αναφερθήκαμε στο προηγούμενο κεφάλαιο. Πιο αναλυτικά:

Έχει δημιουργηθεί ένας πίνακας ο οποίος απεικονίζει κάθε εγγραφή δεδομένων που πραγματοποιείται στην βάση δεδομένων η οποία διαθέτει τα εξής 5 στοιχεία:

1. **ID :** Μοναδικός αριθμός για κάθε εγγραφή δεδομένων που πραγματοποιείται
2. **Timestamp:** Ορίζει την ημερομηνία και την ώρα που πραγματοποιήθηκε η λήψη των δεδομένων από τους αισθητήρες του Arduino.
3. **Load:** Βάρος
4. **Temperature:**Θερμοκρασία
5. **Humidity:**Υγρασία



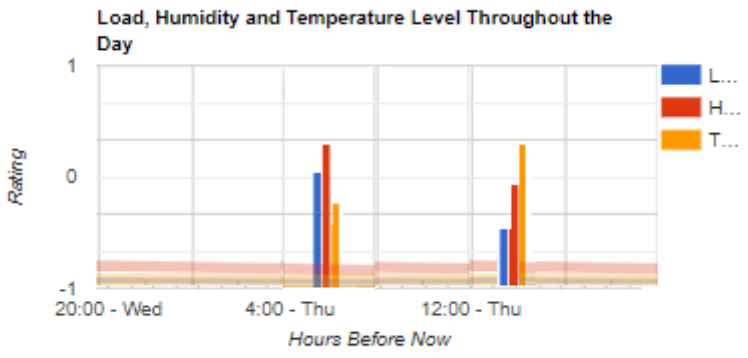
LOAD, HUMIDITY, TEMPERATURE

ID	Timestamp	Load	Temperature	Humidity
593	2018-11-27 21:35:29	15	17	64
592	2018-11-27 21:32:10	16	17	64
591	2018-11-27 21:28:53	15	17	64
590	2018-11-27 21:25:36	15	17	64
589	2018-11-27 21:22:19	15	17	64
588	2018-11-27 21:19:02	15	17	64
587	2018-11-27 21:15:45	15	17	64
586	2018-11-27 21:12:29	16	17	64
585	2018-11-27 21:09:12	17	17	64
584	2018-11-27 21:05:55	18	17	64
583	2018-11-27 21:02:38	19	17	64
582	2018-11-27 20:59:21	19	17	64
581	2018-11-27 20:56:04	20	17	64
580	2018-11-27 20:52:47	20	17	64
579	2018-11-27 20:49:31	21	17	64
578	2018-11-27 20:46:14	20	17	64
577	2018-11-27 20:42:57	19	17	64
576	2018-11-27 20:39:40	17	17	64
575	2018-11-27 20:36:23	16	17	64
574	2018-11-27 20:33:07	16	17	64
573	2018-11-27 20:29:50	15	17	63
572	2018-11-27 20:26:33	15	17	63
571	2018-11-27 20:23:16	15	17	63
570	2018-11-27 20:19:59	15	17	62

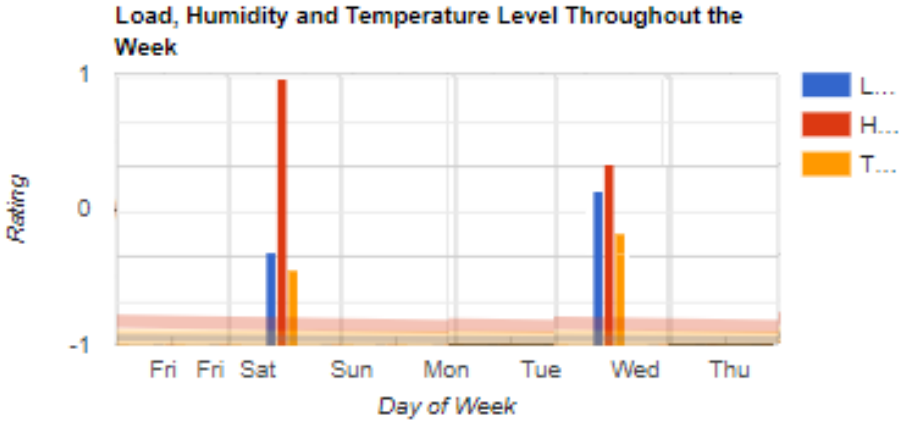
Εικόνα 11. Πίνακας εγγραφών αισθητήρων.

Μέσω όλων των εγγραφών που εμφανίζονται στην παραπάνω εικόνα δημιουργούνται τα γραφήματα που εμφανίζονται στην σελίδα τα οποία χωρίζονται σε 5 κατηγορίες:

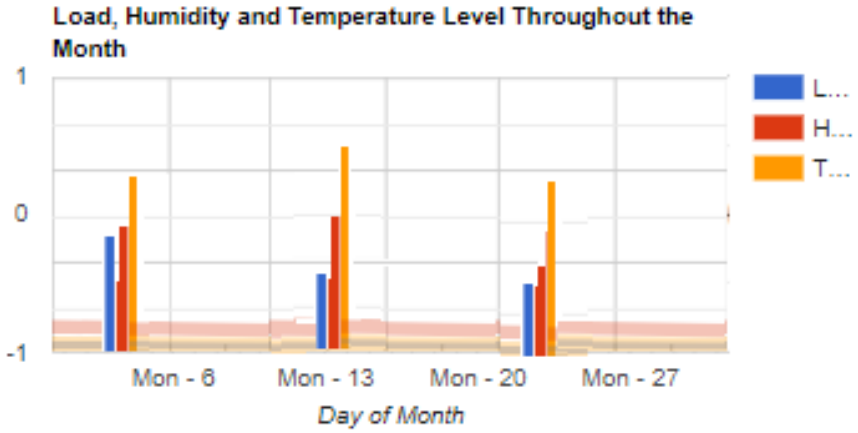
1. Γράφημα ανά 8 ώρες
2. Γράφημα ανά ημέρα
3. Γράφημα ανά εβδομάδα
4. Γράφημα με τον μεγαλύτερο μέσο όρο ανά εβδομάδα του χρόνου
5. Γράφημα με τον μεγαλύτερο μέσο όρο ανά μήνα τον χρόνο



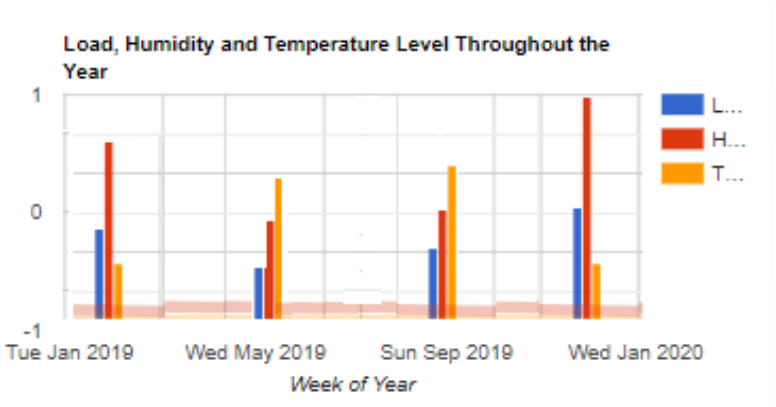
Εικόνα 12. Γράφημα αποτελεσμάτων ανά 8 ώρες



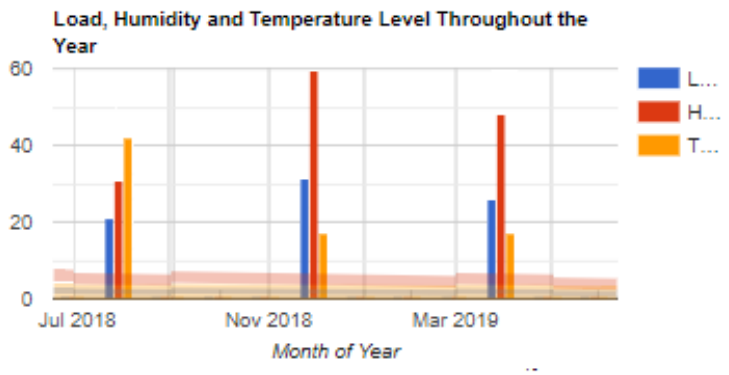
Εικόνα 13. Γράφημα αποτελεσμάτων ανά ημέρα



Εικόνα 14. Γράφημα αποτελεσμάτων ανά εβδομάδα



Εικόνα 15. Γράφημα με τον μεγαλύτερο μέσο όρο ανά εβδομάδα του χρόνου



Εικόνα 16. Γράφημα με τον μεγαλύτερο μέσο όρο ανά μήνα τον χρόνο

4.2 Προβλήματα που παρουσιάστηκαν κατά την υλοποίηση

Όπως είναι φυσικό σε κάθε κατασκευή και δε όταν υπάρχει και μια πολύπλοκη συνδεσμολογία παρουσιάζονται διάφορα προβλήματα. Στην συνέχεια θα αναφερθούμε στα κύρια προβλήματα που αντιμετωπίστηκαν.

4.2.1 Λανθασμένες μετρήσεις βάρους

Όπως προαναφέρθηκε και στο κεφάλαιο 3 και υποενότητα 3.5 κάθε αισθητήρας βάρους έχει τον δικό του συγκεκριμένο αριθμό για το καλιμπράρισμα αλλά ακόμα και δύο ίδιοι αισθητήρες έχουν διαφορετικό. Στην προκειμένη περίπτωση μέχρι να βρεθεί ο σωστός αριθμός υπήρξαν πολλές δοκιμές με διάφορους αριθμούς μέχρις ότου να βρεθεί ο κατάλληλος. Το πρόβλημα όμως δεν σταμάτησε εκεί, κάθε φορά που ενεργοποιείται ο αισθητήρας για ένα διάστημα 10 λεπτών τα αποτελέσματα που δίνει δεν είναι πάντοτε σωστά διότι ο χρόνος που χρειάζεται για να πραγματοποιηθεί το καλιμπραρισμά του δεν είναι σταθερό.

Το ίδιο πρόβλημα παρουσιάστηκε μετά από μερικές ώρες λειτουργίας όπου οι μετρήσεις ήταν λανθασμένες.

4.2.2 Σύνδεση/Αποσύνδεση αισθητήρα βάρους

Για την ορθή λειτουργία του αισθητήρα βάρους χρησιμοποιείται η πλακέτα HX711. Η πλακέτα δεν διαθέτει σταθερά pin έτσι ώστε να είναι εύκολη η σύνδεση των καλωδίων σε αυτήν. Σε αυτή την περίπτωση ένα πρόβλημα που αντιμετωπίστηκε ήταν ότι οι συνδέσεις δεν ήταν σταθερές και έτσι σε ανύποπτο χρόνο ενώ ο αισθητήρας βάρους ήταν συνδεδεμένος και μετέδιδε δεδομένα σταμάταγε. Σε αυτή την περίπτωση εάν γινόταν κάποια κίνηση σε ένα από τα καλώδια αμέσως ο αισθητήρας συνέχιζε να μεταδίδει.

Το πρόβλημα αυτό αντιμετωπίστηκε με την βοήθεια ενός breadboard που χρησιμοποιήθηκε έτσι ώστε οι συνδέσεις που πραγματοποιούνται να είναι πιο σταθερές και αξιόπιστες.

4.2.3 Αδυναμία μετάδοσης δεδομένων με το GsmShield

Στην προσπάθεια μετάδοσης των δεδομένων από τους αισθητήρες στον server αντιμετωπίστηκε ένα πρόβλημα το οποίο για αρκετό καιρό δεν μπορούσε να λυθεί. Ενώ όλες οι συνδέσεις είχαν πραγματοποιηθεί σωστά, οι παράμετροι για την ενεργοποίηση του gsm δικτύου ήταν σωστές μόλις ξεκινούσε η συσκευή μετά από κάποια δευτερόλεπτα απενεργοποιούταν ή η φωτεινή ένδειξη STAUTS και NET έσβηναν με αποτέλεσμα να μην λειτουργεί η συσκευή.

Μετά από μια έρευνα στο διαδίκτυο στην επίσημη σελίδα του GSM Module στις προδιαγραφές αναφερόταν πως για την ορθήλειτουργία απαιτείται 5V και 700maέως 1A. Η συσκευή χρειάζεται 1,7A το ελάχιστο για να μπορέσει να λειτουργήσει σωστά και να πραγματοποιήσει την σύνδεση στο διαδίκτυο για την αποστολή των δεδομένων.

5. Συμπεράσματα & Μελλοντική εξέλιξη

5.1 Συμπεράσματα

Μετά το πέρας αυτής της πτυχιακής τα συμπεράσματα που βγήκαν ήταν πως ο μικροελεγκτής Arduino είναι μια πολύ εύχρηστη πλατφόρμα με πολλές προοπτικές και δυνατότητα πολλαπλών χρήσεων. Η επιλογή να χρησιμοποιηθεί το Arduinoπροήλθε κυρίως από 3 βασικούς παράγοντες:

1. Οικονομική αγορά προϊόντος
2. Ευχέρεια χρήσης με πολλές δυνατότητες
3. Πλούσιο υλικό και παραδείγματα στο διαδίκτυο

Η δημιουργία οποιαδήποτε κατασκευής με την πλατφόρμα του Arduino μπορεί να είναι ταυτόχρονα και πολύ απλή αλλά και αντίστοιχα πολύ δύσκολη και εξειδικευμένη. Η απλότητα που παρέχει την δημιουργία κατασκευών με όλα τα παρελκόμενα που διαθέτει όπως είναι ένα μεγάλο εύρος από αισθητήρες , επεκτάσεις αλλά και οθόνες απεικόνισης δίνει ένα μεγαλύτερο κίνητρο να ασχοληθείς με αυτό. Θα πραγματοποιούσα την ίδια επιλογή για το Arduino ξανά.

5.2 Μελλοντική εξέλιξη

Η έξυπνη μονάδα μετρήσεων που δημιουργήθηκε είναι πλήρως λειτουργική και εξυπηρετική ως προς τον χρήστη. Μελλοντικά θα μπορούσαν να πραγματοποιηθούν κάποιες διαφοροποιήσεις έτσι ώστε να είναι ακόμα πιο λειτουργική αλλά και πιο «έξυπνη». Οι προοπτικές που θα μπορέσουν να πραγματοποιηθούν είναι οι εξής:

- Δημιουργία εφαρμογής για τα κινητά/tablet (APK, IOS) έτσι ώστε ο χρήστης να μπορεί να ανοίγει κατευθείαν την εφαρμογή και μέσα από ένα ειδικά διαμορφωμένο menu να έχει την δυνατότητα να επιλέγει ποια κατηγορία από τις καταγραφές επιθυμεί να δεί.
- Προσθήκη GPS σε περίπτωση που κλαπεί να υπάρχει η δυνατότητα εντοπισμού
- Προσθήκη αισθητήρα φωτιάς για την έγκαιρη ειδοποίηση σε περίπτωση φωτιας
- Προσθήκη φωτοβολταϊκων πάνελ για την πλήρη εξοικονόμηση ενέργειας και αυτονομίας της συσκευής.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Διαδίκτυο των πραγμάτων. Ανακτήθηκε 5 Μαΐου 2019 από, https://el.wikipedia.org/wiki/%CE%94%CE%B9%CE%B1%CE%B4%CE%AF%CE%BA%CF%84%CF%85%CE%BF_%CF%84%CF%89%CE%BD_%CF%80%CF%81%CE%B1%CE%B3%CE%BC%CE%AC%CF%84%CF%89%CE%BD

2. Arduino. Ανακτήθηκε 15 Μαΐου 2019 από , https://el.wikipedia.org/wiki/Arduino

3. Arduino Uno REV3. Ανακτήθηκε 17 Μαΐου 2019 από, https://store.arduino.cc/arduino-uno-rev3

4. What is arduino? Ανακτήθηκε 29 Μαΐου 2019 από, https://learn.sparkfun.com/tutorials/what-is-an-arduino/all

5. Arduino Products. Ανακτήθηκε απο, https://www.arduino.cc/en/Main/Products

6. How to setup the DHT11 Humidity Sensor on an Arduino. Ανακτήθηκε 2 Ιουνίου 2019 από, http://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-an-arduino/

7. Arduino Based Digital Weight scale with loda cell. Ανακτήθηκε 4 Ιουνίου 2019 από ,http://engineerexperiences.com/arduino-based-digital-weight-scale-with-load-cell.html

8. Getting started with the Arduino GSM Shield . Ανακτήθηκε από, https://www.arduino.cc/en/Guide/ArduinoGSMShield