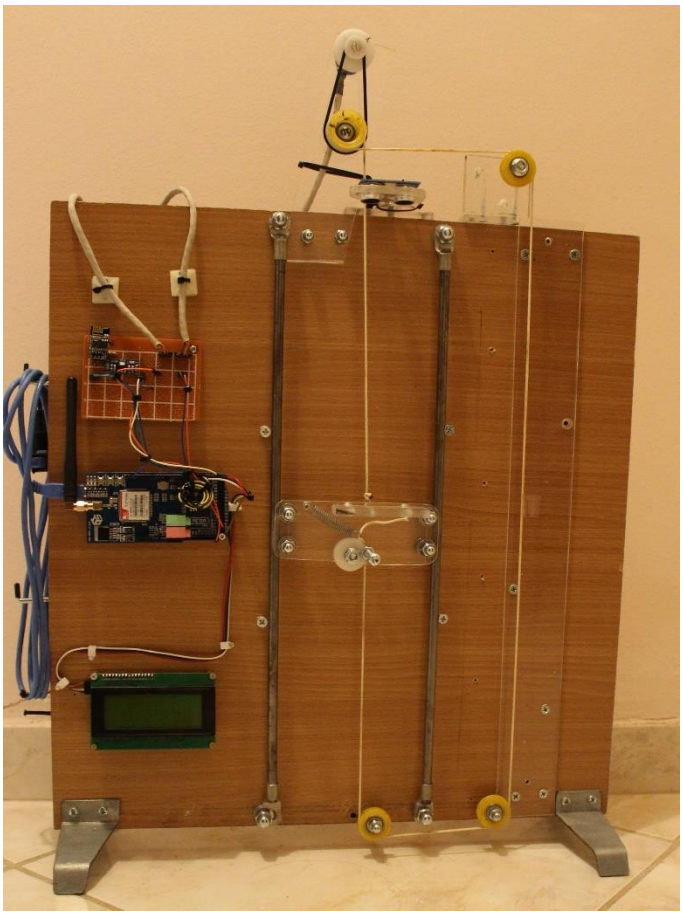
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΑΥΤΟΜΑΤΙΣΜΟΥ Τ.Ε.
ΑΤΕΙ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ελεγκτής δεξαμενής πετρελαίου με διασύνδεση WI-FI και GSM



Του φοιτητή,

Κωνσταντίνου Άγγελου

Αρ. Μητρώου: 112884

Επιβλέπων καθηγητής,

Δρ. Νικόλαος Νικολαΐδης

Θεσσαλονίκη 2018

Ευχαριστίες,

χωρίς την παρουσία, την υποστήριξη και την ανεκτικότητα κάποιων ανθρώπων δεν θα ήταν δυνατή η υλοποίηση της πτυχιακής εργασίας.

Πρώτα από όλους θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή Δρ. Νικόλαο Νικολαΐδη καθηγητή εφαρμογών στο τμήμα Αυτοματισμού του ΑΤΕΙ Θεσσαλονίκης, για την επιστημονική και ηθική υποστήριξη που μου παρείχε καθ' όλη την διάρκεια εκπόνησης της πτυχιακής εργασίας.

Θα ήθελα να ευχαριστήσω τους γονείς μου Άγγελο Πέτρο και Πρωτοπούλου Αλεξάνδρα για την οικονομική και την ηθική στήριξη που μου παρείχαν κατά τη διάρκεια εκπόνησης της εργασίας. Ιδιαίτερη δε μνεία αξίζει στον πατέρα μου διότι με βοήθησε με της απόψεις του πάνω στην μηχανολογία και την μέτρηση των υγρών που βοήθησαν στην εκπόνηση της πτυχιακής εργασίας.

ΠΡΟΛΟΓΟΣ

Ο άνθρωπος θέλοντας να ικανοποιήσει βασικές ανάγκες όπως η θέρμανση ή η πρόσβαση σε ηλεκτρική ενεργεία εκεί που δεν φτάνει το δίκτυο ηλεκτροδότησης χρησιμοποίει γεννήτριες ή καυστήρες θέρμανσης. Το κύριο καύσιμο που τροφοδοτεί τέτοια μηχανήματα είναι το πετρέλαιο το οποίο αποθηκεύεται σε δεξαμενές. Βέβαια οποιοδήποτε υγρό καύσιμο είναι ιδιαίτερα εύφλεκτο και έχει έντονη οσμή, συνεπώς η αποθήκευση του και η λειτουργία των μηχανημάτων που τροφοδοτεί γίνεται σε διαφορετικό χώρο μακριά από τον χώρο άμεσης διαβίωσης ανθρώπων.

Αυτό παρουσιάζει δύο σημαντικά προβλήματα. Πρώτο και σημαντικότερο είναι η έλλειψη ασφάλειας σε πιθανότητα διαρροής του καυσίμου είναι ιδιαίτερα επικίνδυνη λόγω της ίδιας της φύσης του εύφλεκτου υγρού. Δεύτερο πρόβλημα που παρουσιάζεται μέχρι στιγμής είναι η μέτρηση της ποσότητας του υγρού που υπάρχει στη δεξαμενή, δεδομένου ότι οι μέχρι τώρα μηχανικοί τρόποι είναι ανακριβείς , αλλά και ότι η ίδια η δεξαμενή βρίσκεται σε απομακρυσμένη περιοχή, όπου δεν υπάρχει άμεση πρόσβαση.

Στην επίλυση των συγκεκριμένων προβλημάτων έχει επικεντρωθεί η παρούσα πτυχιακή εργασία, κατασκευάζοντας έναν ελεγκτή για την παρακολούθηση της στάθμης μίας δεξαμενής πετρελαίου με γνώμονα την ασφάλεια και την απομακρυσμένη ενημέρωση για την στάθμη.

Το σύστημα βασίζεται στον μικροελεγκτή της ATMEL ATmega2560, η γλώσσα προγραμματισμού που βοήθησε στην υλοποίηση είναι η C++ μέσω του Arduino IDE.

Για την ακριβή μέτρηση της στάθμης χρησιμοποιήθηκε ένα ποτενσιόμετρο μαζί με μία μηχανική συστοιχία από ράουλα και μονάδα μέτρησης αποστάσεων μέσω υπερήχων HC - SR04. Για την απομακρυσμένη αποστολή δεδομένων μέσω δικτύου κινητής τηλεφωνίας επιλέχθηκε το Arduino GSM/GPRS Shield της εταιρίας Tinyshine βασισμένο στο SOC SIM900. Ενώ για την αποστολή των δεδομένων μέσω wifi επιλέχθηκε ο μικροελεγκτής ESP-8266 της εταιρίας Ai-Thinker. Επίσης χρησιμοποιείται και μία οθόνη LCD με σκοπό την απευθείας τοποθέτηση της πάνω στην δεξαμενή για την εμφάνιση χρήσιμων πληροφορίων.

ΠΕΡΙΛΗΨΗ

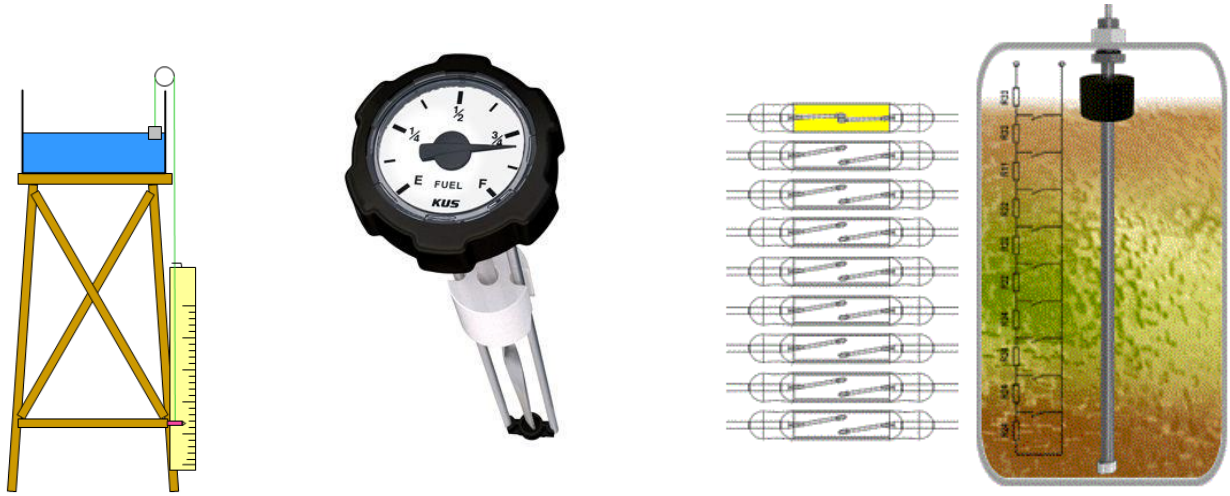
Εισαγωγή

Το πετρέλαιο είναι το πιο διαδομένο καύσιμο που χρησιμοποιείται στην Ελλάδα, ιδιαίτερα εκτός των αστικών κέντρων, κυρίως για θέρμανση ή παραγωγή ηλεκτρικού ρεύματος με γεννήτρια. Η Αποθήκευση του γίνεται σε δεξαμενές πλαστικές ή μεταλλικές.

Ο συνεχής έλεγχος της στάθμης είναι ιδιαίτερα σημαντικός εφόσον έτσι μπορεί κανείς ταυτόχρονα να σιγουρευτεί για την επάρκεια της ποσότητας καυσίμου που έχει και να ελέγξει για τυχόν διαρροές αυτού του εύφλεκτου και καταστροφικού για το περιβάλλον υγρού καυσίμου.

Τρόποι μετρήσεως

Στις οικίες ή μικρές βιομηχανίες οι τρόποι μετρήσεως της στάθμης των δεξαμενών πετρελαίου γίνονται κυρίως με μηχανικού τρόπους.



Εικόνα 1 Τρόποι μέτρησης στάθμης.

Χρησιμοποιώντας φλοτέρ που επιπλέει λόγω της πυκνότητας του υγρού καυσίμου σε συνδυασμό με ράουλα και βαρίδια ή κινώντας έναν ραβδωτό άξονα μπορεί κανείς να προβεί σε μετρήσεις, οι οποίες δυστυχώς δεν είναι ακριβείας και κατ' επέκταση δεν μπορεί να είναι γνωστή η ακριβής ποσότητα του καυσίμου αλλά μόνο εικασίες μπορούν να γίνουν. Με τη χρήση ηλεκτρομηχανολογικών μέσων όπως μαγνητικό φλοτέρ και με μία συστοιχία αντιστάσεων μπορεί το σύστημα μετρήσεων να μετατρέψει την πληροφορία σε ηλεκτρικό ρεύμα καθιστώντας την εύκολη προς αποστολή αλλά και πάλι δεν υπάρχει ακρίβεια στην μέτρηση.

Μπορεί κανείς εύκολα να παρατηρήσει όμως ότι κανένα από τα προαναφερόμενά συστήματα δεν έχει απομακρυσμένο τρόπο ενημέρωσης για την ποσότητα καυσίμου.

Σκοπός της εργασίας

Αυτά τα προβλήματα οδήγησαν στην ανάπτυξη ενός ολοκληρωμένου ελεγκτή που θα ελέγχει συνεχώς την στάθμη του πετρελαίου και θα αποστέλλει τα δεδομένα δίνοντας τη δυνατότητα απομακρυσμένης ενημέρωσης για την ποσότητα καυσίμου που υπάρχει στην δεξαμενή με ακρίβεια, χρησιμοποιώντας δύο διαφορετικούς τρόπους μέτρησης της στάθμης. Επίσης παρέχει τη δυνατότητα αποστολής των στοιχείων μέσω δύο διαφορετικών τρόπων, τοπικά μέσω του δικτύου wifi και απομακρυσμένα μέσω του δικτύου κινητής τηλεφωνίας.

Περιεχόμενα

Ευρετήριο εικόνων και σχημάτων

ΕΙΣΑΓΩΓΗ

Κύριος στόχος της παρούσας πτυχιακής εργασίας είναι η ανάπτυξη και υλοποίηση ενός αυτόνομου συστήματος που θα έχει την δυνατότητα:

- Ελέγχου του ύψους της στάθμης και να το μεταφράζει σε ποσοστό πληρότητας.
- Ελέγχου ύπαρξης διαρροής στην δεξαμενή.
- Απομακρυσμένη αποστολή των δεδομένων.

Στο πρώτο κεφάλαιο της εργασίας παρουσιάζεται και αναλύεται η κατασκευή για την μακέτα. Πρώτα σε μηχανικό επίπεδο και στη συνέχεια γίνεται παρουσίαση των ηλεκτρονικών εξαρτημάτων που χρησιμοποιήθηκαν και στο τρόπο που είναι συνδεδεμένα μεταξύ τους.

Στο δεύτερο κεφάλαιο της εργασίας, εξετάζεται η λειτουργία του κώδικα και αναλύονται τα σημαντικότερα βήματα αυτού.

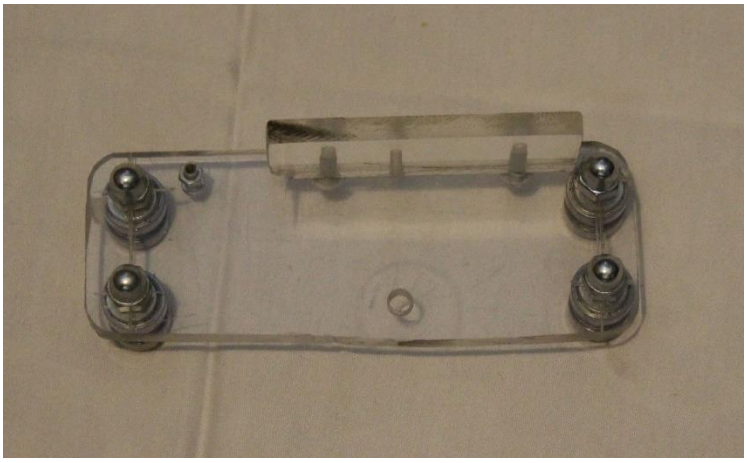
Και τέλος στο τρίτο κεφάλαιο παρουσιάζονται τα συμπεράσματα και οι μελλοντικές πιθανές προσθήκες που μπορούν να γίνουν σε μία τέτοια κατασκευή.

1. Κατασκευή.

1.1 Μηχανική ανάλυση της κατασκευής.

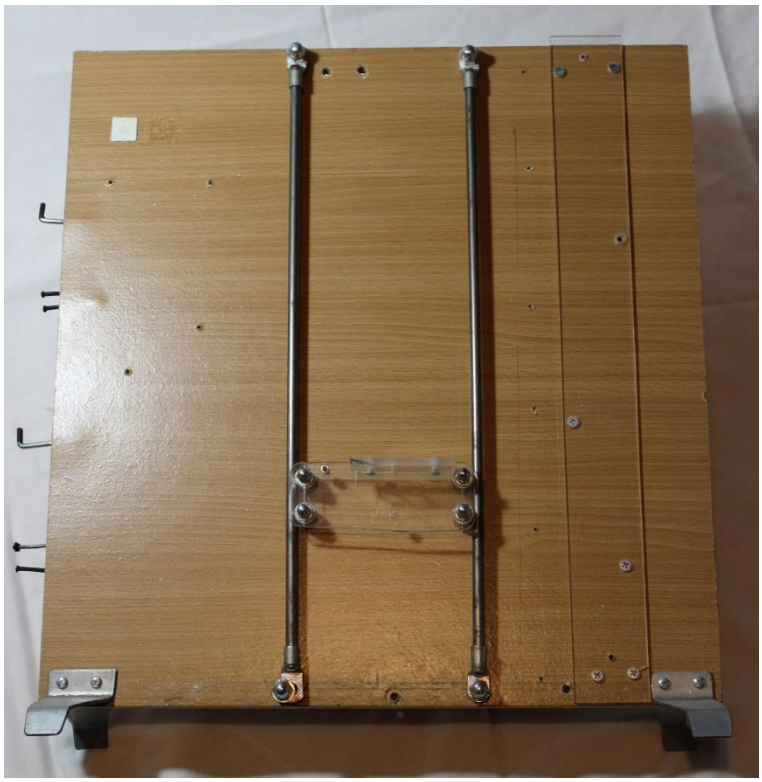
Ένα από τα κυριότερα προβλήματα που προέκυψε κατά την ανάπτυξη της πτυχιακής εργασίας ήταν η αναπαράσταση της στάθμης μίας δεξαμενής, για την δοκιμή και τον έλεγχο λειτουργίας του ελεγκτή σε εργαστηριακό περιβάλλον. Η λύση σε αυτό το πρόβλημα είναι η κατασκευή ενός πρωτότυπου μηχανικού συστήματός που θα μπορεί να προσομοιώνει την στάθμη σε μικρή κλίμακα και χωρίς την χρήση υγρού για να είναι ασφαλή στις δοκιμές, επίσης να είναι χειροκίνητο και να έχει δυνατότητα άμεσης εναλλαγής της στάθμης.

Η κατασκευή αυτή βασίζεται σε μία πλατφόρμα που μπορεί μόνο να κινηθεί ελεύθερα στον κάθετο άξονα της, με αυτόν τον τρόπο μπορεί να προσομοιωθεί η στάθμη μίας δεξαμενής και κατ' επέκταση το φλοτέρ που επιπλέει στην επιφάνεια του υγρού.



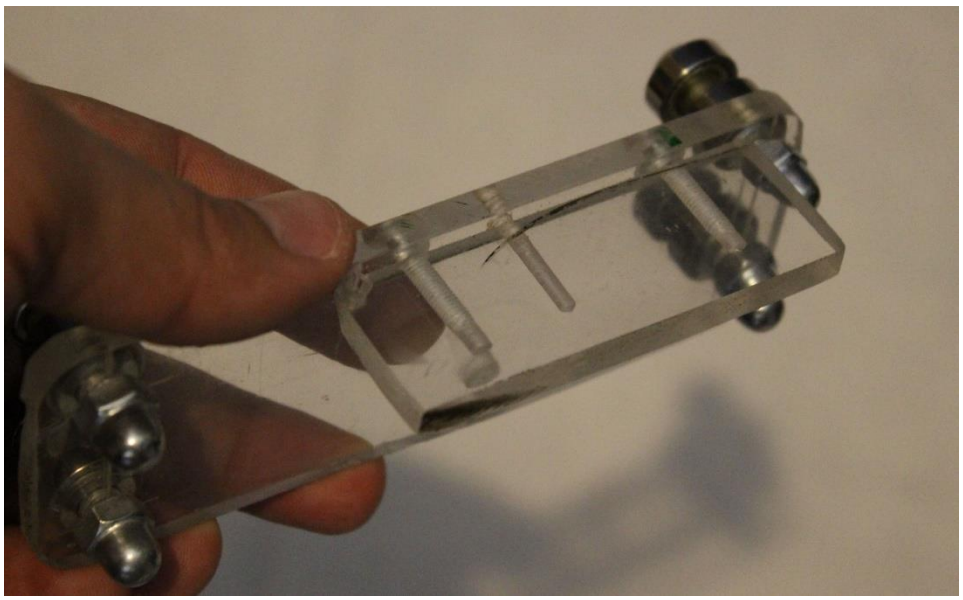
Εικόνα 2 Πλατφόρμα προσομοίωσης στάθμης (1)

Σε ένα τεμάχιο ξύλου 47.5 X 49.5 εκατοστών και πάχους 1.7 εκατοστών τοποθετήθηκαν δύο μεταλλικοί ράβδοι μήκους 46.5 εκατοστών όπου κατά μήκος τους θα μπορεί να κινείται η πλατφόρμα.



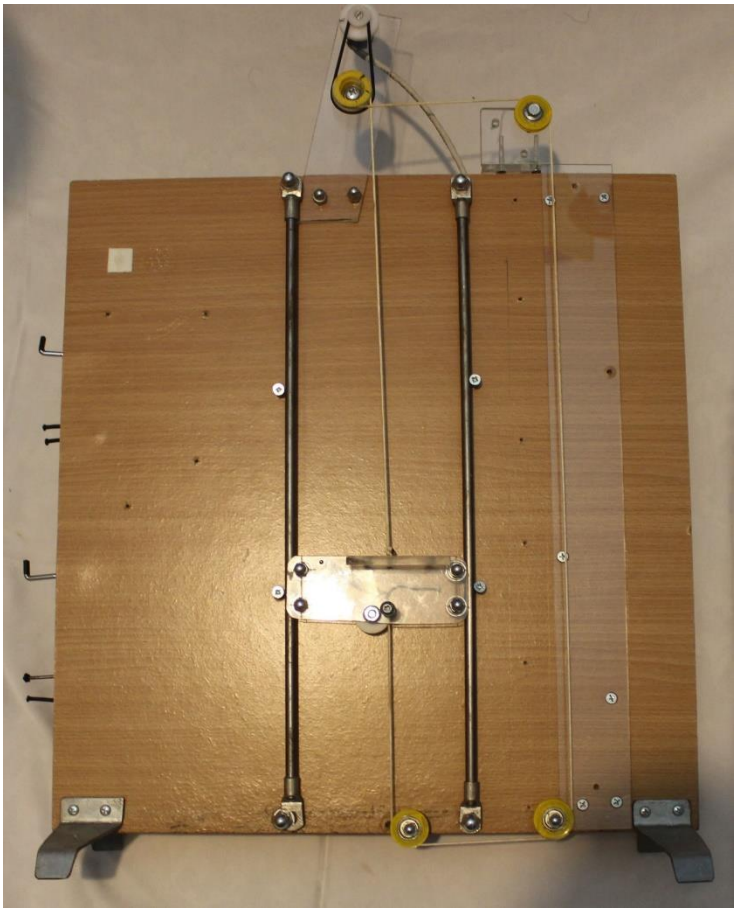
Εικόνα 3 Σύστημα προσομοίωσης στάθμης (1)

Για την κατασκευή της πλατφόρμας χρησιμοποιήθηκε ένα κομμάτι Plexiglas 13 X 5 εκατοστών και πάχους 0.5 εκατοστών επάνω στο οποίο τοποθετήθηκαν 4 συστήματα κύλισης, τα οποία προσαρμόστηκαν για την σωστή κύλιση της πλατφόρμας πάνω σε δύο μεταλλικές ράβδους.

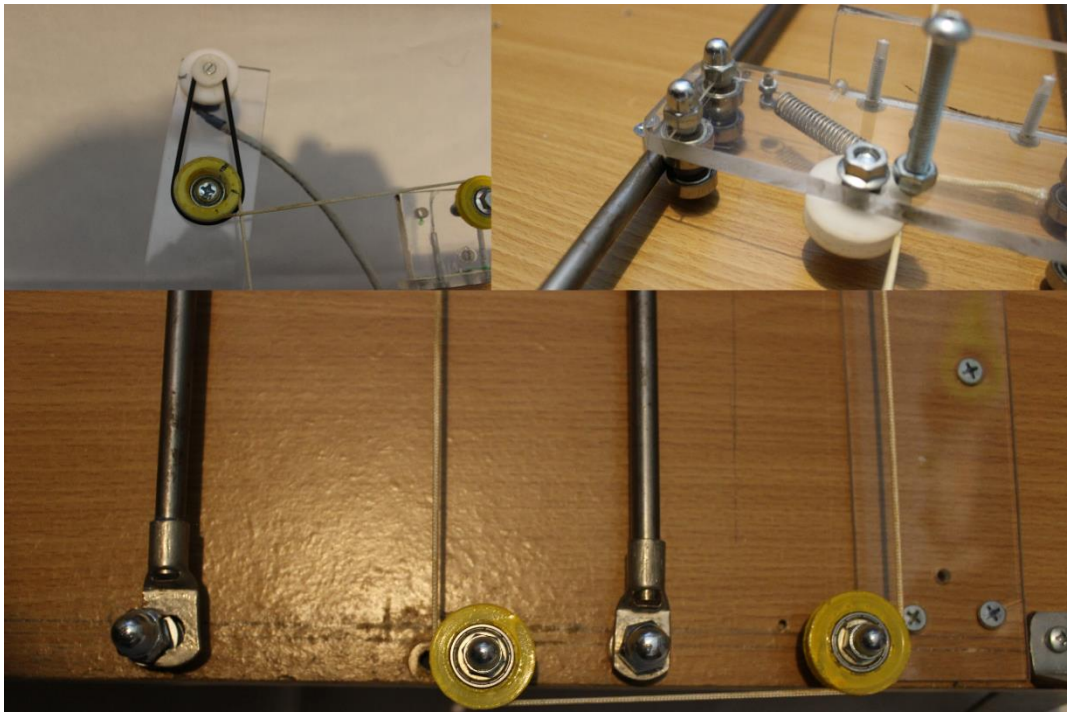


Εικόνα 4 Πλατφόρμα προσομοίωσης στάθμης (2)

Για την διατήρηση της πλατφόρμας πάνω στις ράβδους σε οποιοδήποτε σημείο δημιουργήθηκε ένα κλειστό κύκλωμα από σπάγκο και ράουλα που προσαρμόστηκε πάνω στο μοντέλο.

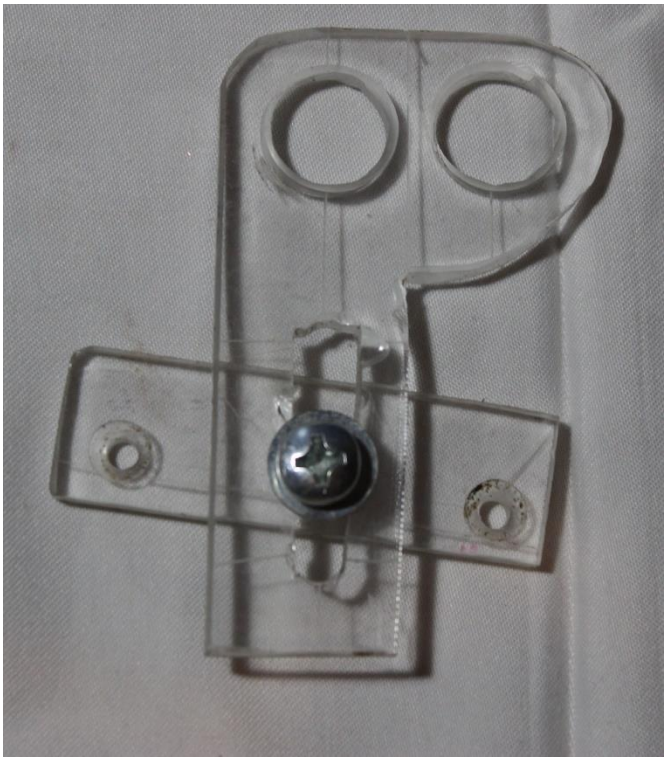


Εικόνα 5 Σύστημα προσομοίωσης στάθμης (2)

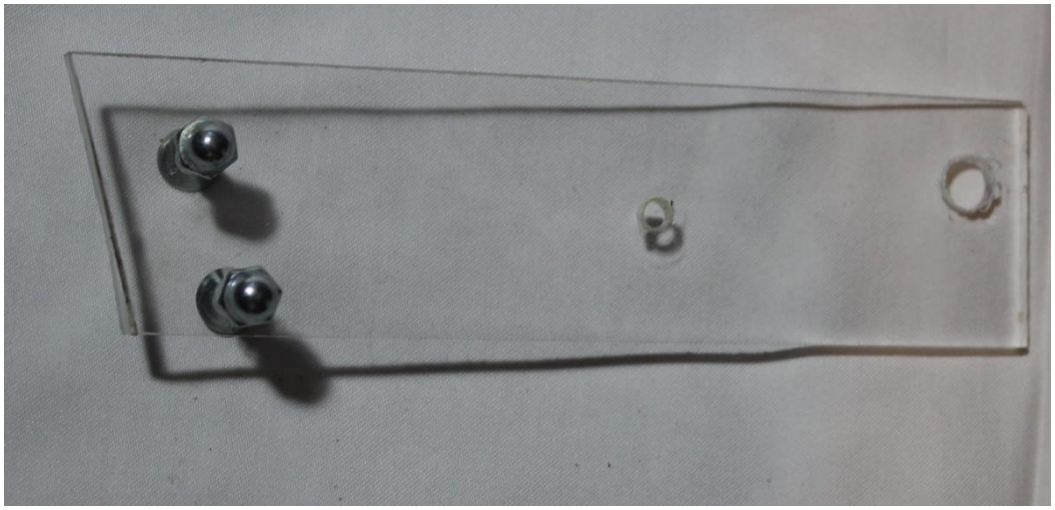


Εικόνα 6 Ράουλα στο κλειστό σύστημα

Κατασκευάστηκαν και τοποθετήθηκαν βάσεις για τα αισθητήρια που θα τοποθετηθούν πάνω στο μηχανικό σύστημα προσομοίωσης στάθμης.

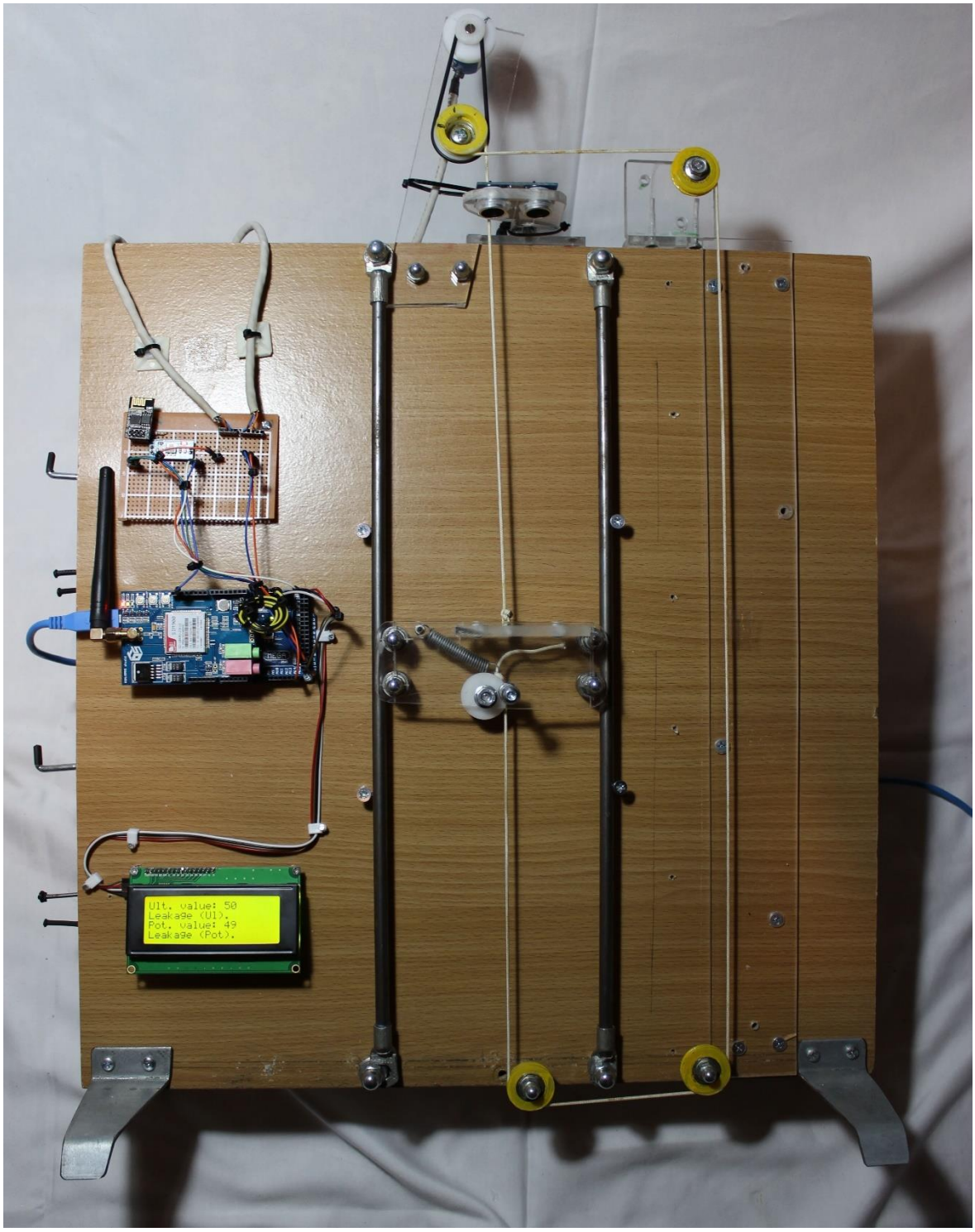


Εικόνα 7 Βάση αισθητήριου υπερήχων.



Εικόνα 8 Βάση αισθητήριου απόστασης.

Τα ηλεκτρονικά στοιχεία του ελεγκτή εγκαταστάθηκαν στο αριστερό κομμάτι, ενώ στα δεξιά τοποθετήθηκε ένας μηχανικός δείκτης ένδειξης της στάθμης.



Εικόνα 9 Σύστημα προσομοίωσης στάθμης (3)

1.2 Ηλεκτρονική ανάλυση της κατασκευής.

Ο ελεγκτής στάθμης αποτελείται από τρία βασικά υποσυστήματα:

- Τη μονάδα ελέγχου.
- Τα αισθητήρια μετρήσεως.
- Τις μονάδες επικοινωνίας.

1.2.1 Μονάδα ελέγχου.

Η μονάδα ελέγχου, η λεγόμενη <<καρδιά>> του συστήματος που φιλοξενεί τον κώδικα και είναι υπεύθυνη για κάθε ενέργεια του συστήματος, αποτελείται από τον μικροελεγκτή Arduino Mega 2560. Ο σκοπός του είναι να λαμβάνει τα δεδομένα από τα αισθητήρια μετρήσεως , να τα επεξεργάζεται και να προβεί στις απαραίτητες ενέργειες για την αποστολή τους προς τον χρήστη μέσω των μονάδων επικοινωνίας. Παρακάτω υπάρχει μία σύντομη περιγραφή του .

Ο μικροελεγκτής Arduino Mega 2560 είναι βασισμένος στον μικροεπεξεργαστή της ATMEL ATmega2560. Αποτελείται από 54 ψηφιακές εισόδους/εξόδους (15 από τις οποίες έχουν τη δυνατότητα παραγωγής σήματος PWM), 16 αναλογικές εισόδους, 4 UARTS, έναν κρύσταλο των 16 MHz, μία διεπαφή USB, μία υποδοχή τροφοδοσίας και έναν ICSP header.

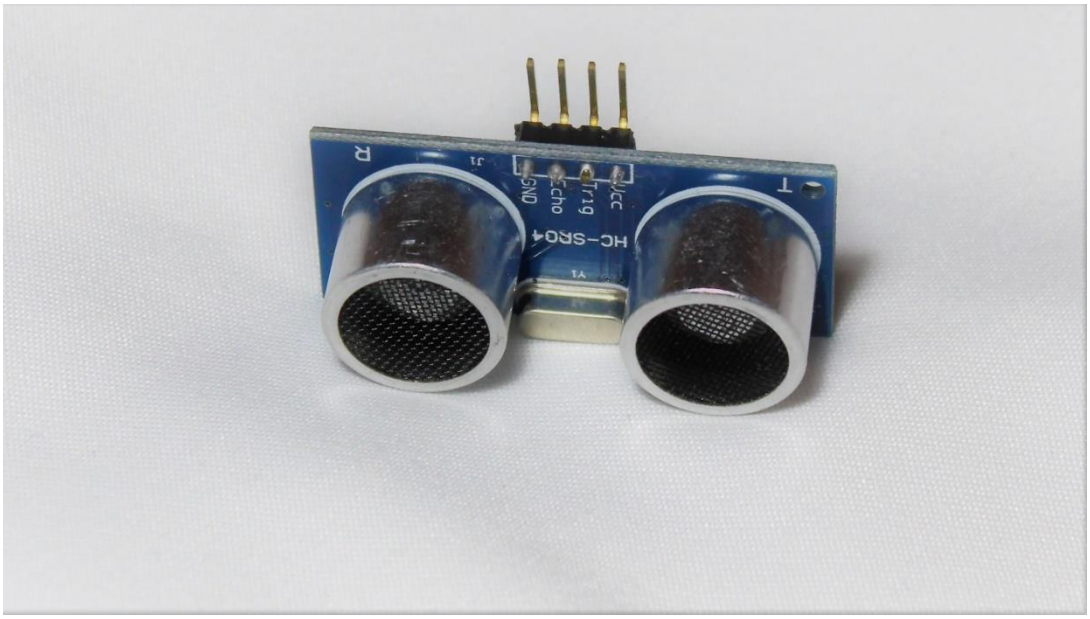
Εικόνα 10 Arduino Mega2560

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
LED_BUILTIN	13
Length	101.52 mm
Width	53.3 mm
Weight	37 g

1.2.2 Αισθητήρια μετρήσεως.

Θεωρήθηκε χρήσιμο για τη μέτρηση της στάθμης να γίνει χρήση δύο αισθητηρίων, τα οποία να μετράνε δύο διαφορετικές παραμέτρους για πιο ακριβή συνολικό αποτέλεσμα. Το ένα αισθητήριο μετράει την απόσταση της στάθμης του υγρού καυσίμου από το αισθητήριο και μέσω αλγόριθμου υπολογίζει την ποσότητα του. Ταυτόχρονα το άλλο αισθητήριο μετράει τη στάθμη γνωρίζοντας πόσο έχει μετακινηθεί το φλοτέρ από το πάτο της δεξαμενής.

Για την μέτρηση της απόστασης έγινε χρήση της μονάδα μέτρησης απόστασης μέσω υπερήχων HC - SR04, που έχει την δυνατότητα να μετρήσει αποστάσεις από 2cm έως 400cm με ακρίβεια της τάξεως των 3mm. Η μονάδα αποτελείται από τον εκπομπό, το δέκτη και το κύκλωμα ελέγχου.



Εικόνα 11 Αισθητήρας υπερήχων HC-SR04

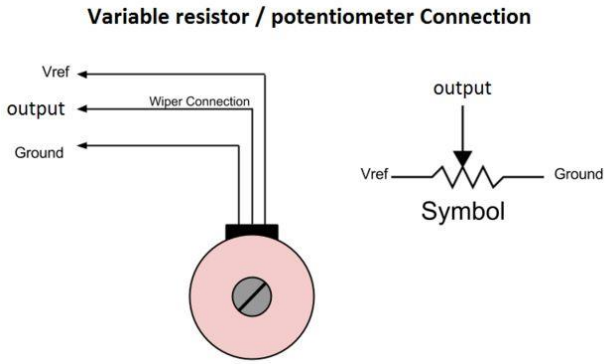
Working Voltage	DC 5 V
Working Current	15mA
Working Frequency	40kHz
Max Range	4m
Min Range	2cm
Measuring Angle	15 degree
Trigger Input Signal	10uS TTL pulse
Echo Output Signal	Input TTL lever signal and the range in proportion
Dimension	45*20*15mm

Η βάση λειτουργίας της μονάδας είναι η εξής:

1. Στέλνεται στο κύκλωμα ελέγχου έναν θετικό παλμό τουλάχιστο 10 us.
2. Η μονάδα στέλνει αυτόματα οκτώ (8) παλμούς των 40KHz μέσω του εκπομπού και ελέγχει πότε θα λάβει ο δέκτης το σήμα πίσω.
3. Με το πέρας της αποστολής των 8 παλμών του δέκτη η σημαία echo γίνεται high μέχρι να λάβει το σήμα ο δέκτης, οπότε η απόσταση μπορεί να υπολογιστεί από την φόρμουλα:

$$\frac{\text{χρόνος high του echo } * \text{ ταχύτητα του ήχου } (340\frac{M}{s})}{2}$$

Για την μέτρηση του μετακίνησης του φλοτέρ χρησιμοποιείται ένα ποτενσιόμετρο σε συνεργασία με το μηχανικό σύστημα του φλοτέρ. Το ποτενσιόμετρο είναι αναλογικό ηλεκτρονικό εξάρτημα, με τρεις ακροδέκτες. Το ποτενσιόμετρο είναι ένας μεταβλητός αντιστάτης που χρησιμοποιείται ως διαιρέτης τάσης, για τη μεταβολή δηλαδή του ηλεκτρικού δυναμικού.



Εικόνα 12 Ποτενσιόμετρο

1.2.3 Μονάδες επικοινωνίας.

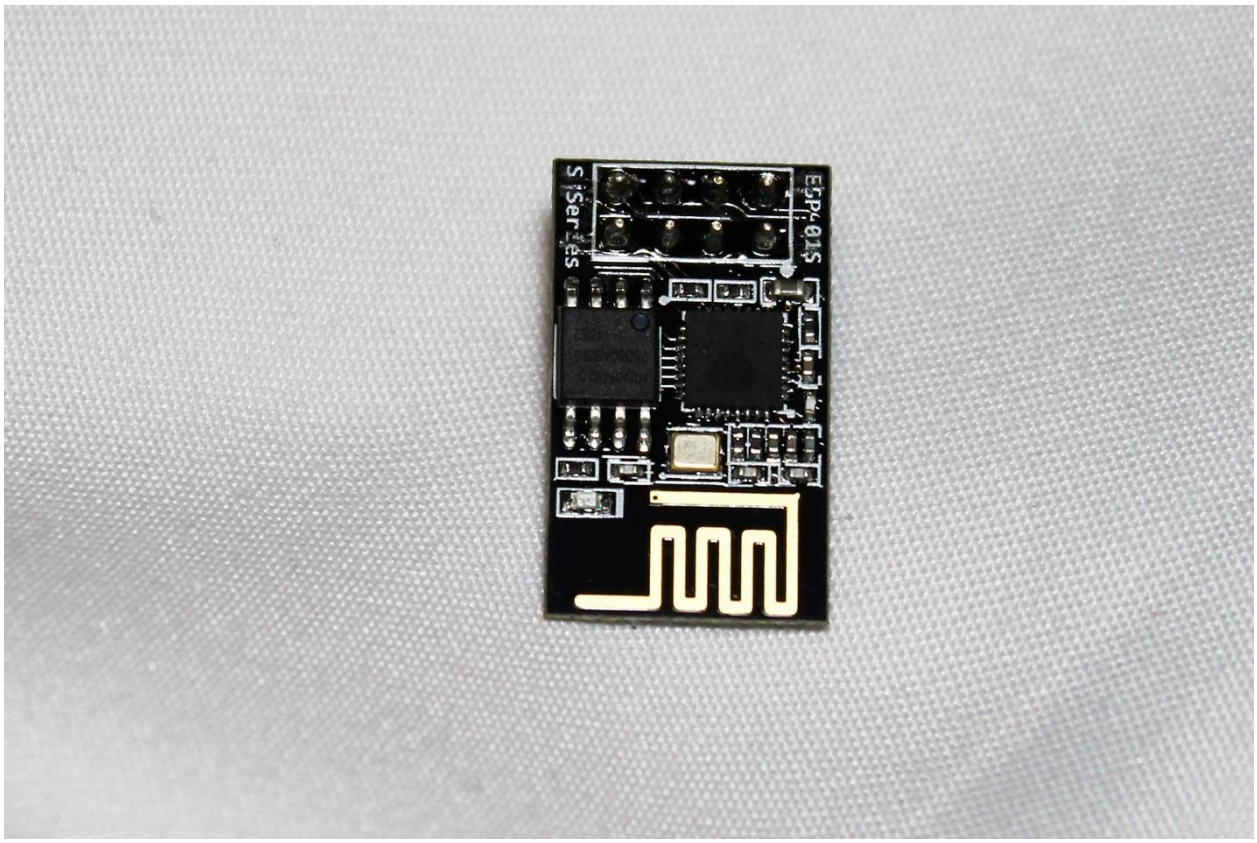
Για την απομακρυσμένη αποστολή πληροφοριών χρησιμοποιούνται δυο διαφορετικές μονάδες για την διασύνδεση σε δίκτυα wifi αναλαμβάνει ο ESP8266, ενώ την διασύνδεση στο δίκτυο κινητής τηλεφωνίας το GSM/GPRS shield για Arduino.

Διασύνδεση wifi.

Η μονάδα ESP8266 είναι ένα SOC με ενσωματωμένα τα πρωτόκολλα TCP/IP που μπορούν να δώσουν τη δυνατότητα σε οποιαδήποτε συσκευής να αποκτήσει σύνδεση σε δίκτυα wifi.

Το ESP8266 μπορεί να φιλοξενήσει οποιαδήποτε πληροφορία και να λειτουργήσει σαν HTTP SERVER σε κάποιο δίκτυο ή να δημιουργήσει το δικό του δίκτυο wifi. Επίσης μπορεί να αποστείλει δεδομένα σε οποιοδήποτε HTTP SERVER.

Στο σύστημα που χρησιμοποιείται το ESP8266 λειτουργεί σαν βοηθητικό υποσύστημα. Εγκαθιδρύει, μέσω του Arduino MEGA 2560, δίκτυο wifi και εν συνεχεία λειτουργεί σαν HTTP SERVER, όπου φιλοξενεί μία σελίδα με τις πληροφορίες της στάθμης.

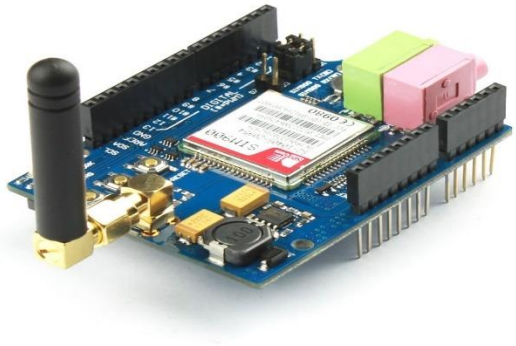


Εικόνα 13 ESP 8266

Διασύνδεση δικτύου κινητής τηλεφωνίας.

Η GPRS/GSM Shield είναι σχεδιασμένη αποκλειστικά για το Arduino λόγω της επακριβούς τοποθέτησης και ενσωμάτωσης της μίας πλακέτας πάνω στην άλλη εξαλείφοντας την ανάγκη διασύνδεσης με εξωτερικά καλώδια.

Παρέχει τη δυνατότητα να αποστέλλονται μέσω του δικτύου κινητής τηλεφωνίας sms ή ακόμα και μετάδοσης φωνής για την άμεση και απομακρυσμένη αποστολή πληροφοριών. Ο τρόπος που έχει επιλεγεί στην παρούσα εργασία είναι το sms.



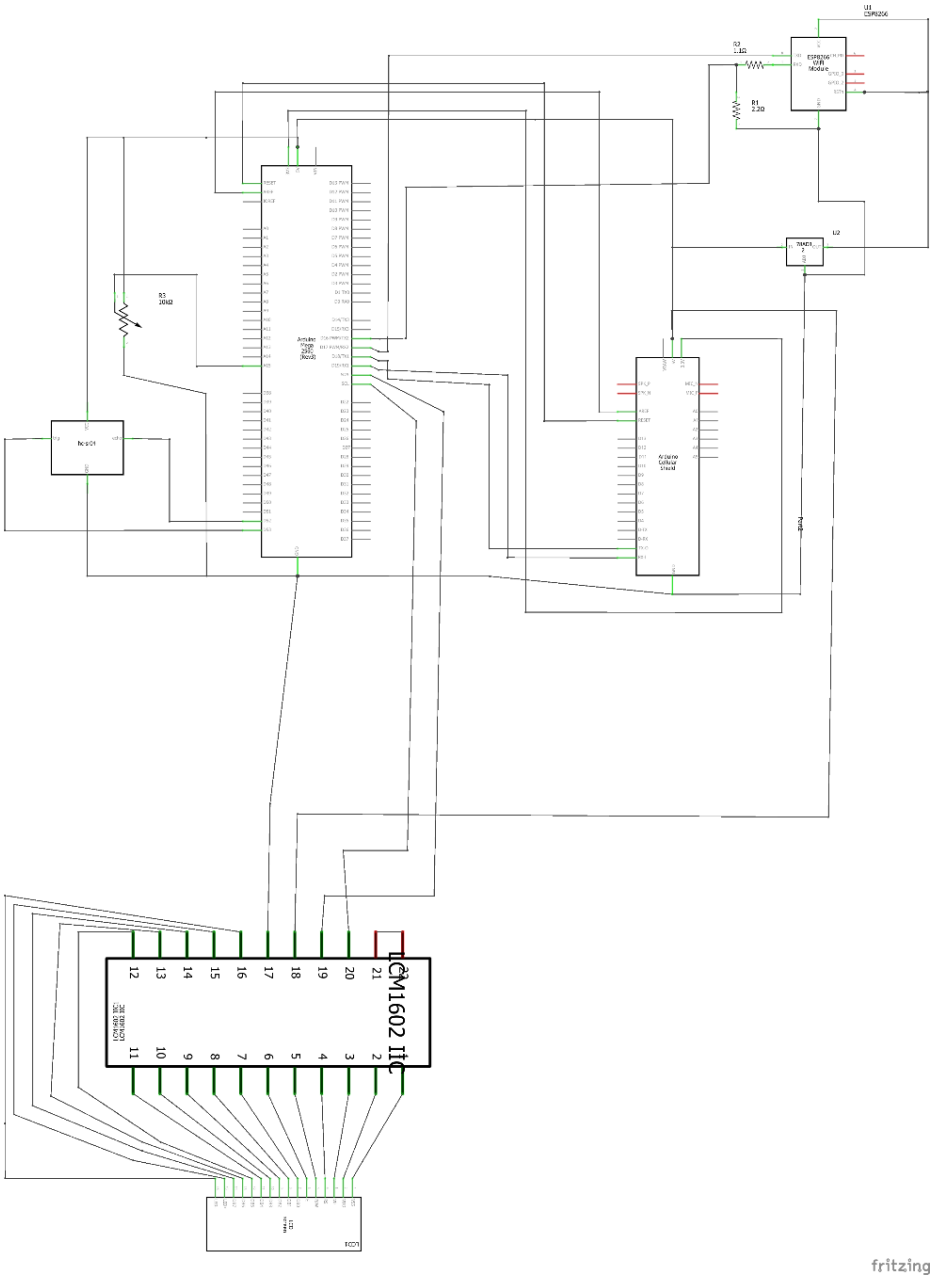
Εικόνα 14 GSM/GPRS Arduino Shield

Στο σύστημα υπάρχει επίσης μία μονάδα LCD που παρέχει την δυνατότητα να απεικονίζεται οποιαδήποτε πληροφορία χρειάζεται στον χρήστη, αποτελείται από την οθόνη 20 χαρακτήρων και 4 γραμμών, και τη μονάδα επικοινωνίας .

Το πρωτόκολλο που χρησιμοποιείται για την επικοινωνία της οθόνης με τον μικροελεγκτή είναι το I2C, έτσι παρέχεται η δυνατότητα διασύνδεσης στο σύστημα έως και 112 στοιχεία δεσμεύοντας μόνο αυτές τις δύο γραμμές, έχοντας επομένως τη δυνατότητα μεταγενέστερης επέκτασης του συστήματος μας χωρίς τη δέσμευση άλλων γραμμών καλωδίων.

Εικόνα 15 LCD 2004A

Το συνολικό ηλεκτρονικό σχέδιο του κυκλώματος παρατίθεται παρακάτω :

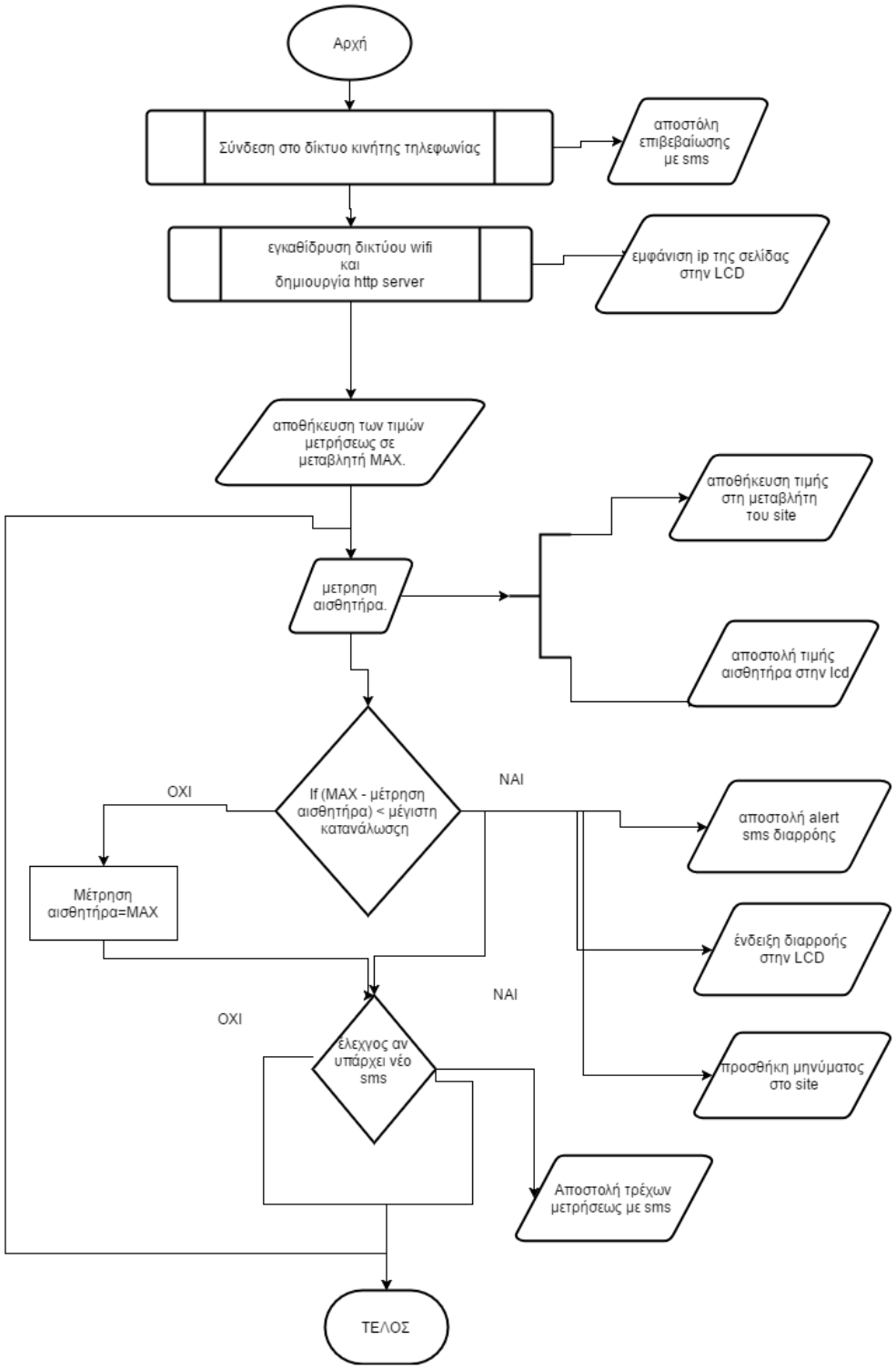


Εικόνα 16 Συνολική απεικόνιση του κυκλώματος

Στο κέντρο του σχεδίου υπάρχει η κεντρική μονάδα επεξεργασίας, στο αριστερό κομμάτι του σχεδίου απεικονίζονται τα αισθητήρια, στο δεξιό μέρος του σχεδίου τα υποσυστήματα επικοινωνίας. Ενώ στο κάτω μέρος η οθόνη LCD μαζί με τα απαραίτητα εξαρτήματα για την σειριακή επικοινωνία.

2. Προγραμματισμός και ανάλυση του κώδικα.

Πρώτα παρατίθεται το διάγραμμα ροής του προγράμματος για να βοηθήσει στην κατανόηση της λειτουργίας του κώδικα.



Εικόνα 17 Διάγραμμα ροής του προγράμματος.

Πιο συγκεκριμένα η σειρά βημάτων λειτουργίας είναι η ακόλουθη:

1. Σύνδεση του ελεγκτή με το δίκτυο κινητής τηλεφωνίας.
2. Εγκαθίδρυση δικτύου wifi και δημιουργία HTTP Server με σκοπό την φιλοξενία του site για την δημοσίευση των μετρήσεων.
3. Μέτρηση αισθητηρίων και αποθήκευση σε μεταβλητές MAX για τον έλεγχο ύπαρξης διαρροής.
4. Μέτρηση των δεδομένων από τους αισθητήρες και αποθήκευση σε μεταβλητές.
5. Γίνεται έλεγχος για διαρροή με την βοήθεια της μεταβλητής MAX. Αν υπάρχει ένδειξη διαρροής γίνεται αυτόματα αποστολή sms και ανανέωση του site.
6. Έλεγχος για το αν υπάρχει sms που ζητάει την αποστολή του επιπέδου της στάθμης. Αποστολή απάντησης εφόσον χρειάζεται.

Στην συνέχεια θα αναλυθεί η λειτουργία του συστήματος :

2.1 Σύνδεση του ελεγκτή με το δίκτυο κινητής τηλεφωνίας.

Για την διασύνδεση του ελεγκτή με το δίκτυο κινητής τηλεφωνίας χρησιμοποιείται η βιβλιοθήκη `SIM900.h` που δημιουργήθηκε ιδικά για την πλατφόρμα του Arduino, για την αποστολή των sms χρησιμοποιείται η βιβλιοθήκη `sms.h`. Ενώ για την επικοινωνία του μικροελεγκτή με το SoC SIM900 χρησιμοποιείται η βιβλιοθήκη `SofwareSerial.h` της πλατφόρμας Arduino IDE.

```
1.    //GSM
2.      #include "SIM900.h"
3.      #include <SoftwareSerial.h>
4.      #include "sms.h"
```

Για την ενεργοποίηση και αρχή λειτουργίας του GSM/GPRS Shield χρησιμοποιείται η εντολή `gsm.begin();`

```
1.    Serial1.begin(9600);
2.      if (gsm.begin(9600))
```

Η αποστολή sms γίνεται μέσω της εντολής `sms.SendSMS(phone_num , msg);`

```
1.   if (sms.SendSMS("+306979532390", "Sent 1 to find level."))
```

Αξίζει να σημειωθεί ότι μια Sim κάρτα έχει αποθηκευτικό χώρο για 20 sms μόνο, συνεπώς, κρίθηκε αναγκαίο όπως προβλεφθεί τρόπος για τη διαγραφή των sms, με την εντολή `sms.DeleteSMS()` που ενεργοποιείται εφόσον γίνει πρώτα ο έλεγχος κατά πόσο υπάρχουν 5 sms αποθηκευμένα στην κάρτα Sim.

```
1.   if (sms_position >= 5)
2.           {
3.             for(int i=1;i<=5;i++)
4.              {
```

```
            sms.DeleteSMS(i);
             }
           }
```

2.2 Εγκαθίδρυση δικτύου wifi και δημιουργία HTTP Server.

Για την επικοινωνία μεταξύ των δύο μικροελεγκτών, του ATMEGA 2560 και του ESP 8266 χρησιμοποιούνται εντολές AT. Οι εντολές AT είναι ένα πακέτο εντολών που δίνονται από τον κατασκευαστή για την άμεση λειτουργία του μικροελεγκτή, έτσι δεν χρειάζεται να γραφτεί ένα ξεχωριστό πρόγραμμα οπού θα φορτωθεί στο ESP 8266.

Η επικοινωνία γίνεται μέσω της δεύτερης σειριακής θύρας του Arduino. Πρώτα πρέπει να γίνει επανεκκίνηση του ESP με την εντολή AT+RST για να απαλείφουν οι προηγούμενες ρυθμίσεις του. Επιλέγεται ο τρόπος λειτουργίας εκπομπής μέσω της εντολής AT+CWMODE, στη συγκεκριμένη περίπτωση σαν Access point. Στο συγκεκριμένο ESP έχει καθοριστεί σαν προεπιλεγμένη διεύθυνση η IP : 192.168.4.1, οπού προβάλεται στην LCD. Με την εντολή AT+CIPMUX παρέχεται η δυνατότητα το ESP να διατηρεί πολλές συνδέσεις μέσω wifi. Τέλος, με την εντολή AT+CIPSERVER δύναται να δημιουργήθει ένα TCP server στο port 80,έτσι ώστε να υπάρχει προσβασιμότητα μέσω του πρωτόκολλου HTTP από οποιοδήποτε browser.

```
  Serial2.begin(115200);
     sendData("AT+RST\r\n",2000,DEBUG); // reset module
     sendData("AT+CWMODE=2\r\n",1000,DEBUG); // configure as access point
     lcd.setCursor(0,2);
     lcd.print("IP : 192.168.4.1");
     sendData("AT+CIFSR\r\n",1000,DEBUG); // get ip address

     //lcd.print(response);
     sendData("AT+CIPMUX=1\r\n",1000,DEBUG); // configure for multiple connections

     sendData("AT+CIPSERVER=1,80\r\n",1000,DEBUG); // turn on server on port 80
```

Για να διαπιστωθεί εάν υπάρχει κάποιο αίτημα, δηλαδή αν κάποιος θέλει να επισκεφτεί το site και για την διαχείριση του τυχόν αιτήματος, δηλαδή την αποστολή του site σε αυτόν που το ζήτησε χρησιμοποιείται ο παρακάτω έλεγχος, εν συντομία ελέγχεται αν ο ESP στέλνει κάποιο δεδομένο μέσω της σειριακής επικοινωνίας. Εάν ληφθεί κάποιο μήνυμα από το ESP στην συνέχεια ελέγχεται εάν υπάρχει η εντολή AT+IPD, που σημαίνει ότι κάποιος ζητάει από το ESP τις πληροφορίες που βρίσκονται στο site. Το μόνο που απομένει στην συνέχεια είναι να αποσταλεί το site σε μορφή γλώσσας HTML.

```
if(Serial2.available()) // check if the esp is sending a message
  {
    if(Serial2.find("+IPD,"))
      {
       delay(1000);

       int connectionId = Serial2.read()-48;
// subtract 48 because the read() function returns
// the ASCII decimal value and 0 (the first decimal number) starts at 48
```

```
        String webpage = "<html><center><img src=http://www.autom.teithe.gr/img/logogr.png><h1>Ptyxiakh ergasia <p>Konstantinou Angelou</h1><p><h2>Potensiometer : ";
              webpage += SensorPot;
              webpage += " +-2 %.</h2><p><h2>Ultrasonic : ";
              webpage += duration;
              webpage += " +-2 %.<br><br>";
              webpage += leakageStatus;
        String cipSend = "AT+CIPSEND=";
        cipSend += connectionId;
        cipSend += ",";
        cipSend +=webpage.length();
        cipSend +="\r\n";

        sendData(cipSend,1000,DEBUG);
        sendData(webpage,1000,DEBUG);
         webpage="<input type=\"button\"name=\"b1\"value=\"Refresh\"onclick=\"location.href='/'\"></center></html>";

     cipSend = "AT+CIPSEND=";
     cipSend += connectionId;
     cipSend += ",";
     cipSend +=webpage.length();
     cipSend +="\r\n";

     sendData(cipSend,1000,DEBUG);
     sendData(webpage,1000,DEBUG);


       String closeCommand = "AT+CIPCLOSE=";
       closeCommand+=connectionId; // append connection id
       closeCommand+="\r\n";

       sendData(closeCommand,3000,DEBUG);
```

2.3 Μέτρηση αισθητήριων και αποθήκευση σε μεταβλητή τύπου MAX.

Η λειτουργία αυτή είναι χρήσιμη, εφόσον δημιουργεί μία δικλείδα ασφαλείας στον ελεγκτή. Γνωρίζοντας τη συνολική κατανάλωση του γενικού συστήματος που υπάρχει και του συνολικού χρόνου λειτουργίας ενός κύκλου του προγράμματος μπορεί κανείς εύκολα με μία απλή σύγκριση της μέτρησης και της μεταβλητής MAX να γνωρίζει εάν υπάρχει παραπάνω μείωση της στάθμης, συνεπώς και διαρροή.

Αυτό επιτυγχάνεται κάνοντας μία πρώτη αποθήκευση των μετρήσεων σε μια μεταβλητή MAX στην έναρξη του προγράμματος.

```
SensorPot = analogRead(15);
    SensorPot = map (SensorPot, 1020, 0, 0, 100);
    PotMax = SensorPot;
```

Και

```
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
```

```
5.      digitalWrite(trigPin, HIGH);
6.      delayMicroseconds(10);
7.      digitalWrite(trigPin, LOW);
8.      duration = pulseIn(echoPin, HIGH);
9.      duration =map(duration ,2389, 239, 0, 100);
10.     durationMax = duration;
```

Εφόσον παρέχεται η τιμή των μεταβλητών `Max`, μπορεί κανείς εύκολα να ελέγξει στο κεντρικό πρόγραμμα με μία απλή σύγκριση.

```
  if ((PotMax - SensorPot)>10)
```

και

```
if ((durationMax - duration)>10)
```

Με το αριθμό 10 να προσομοιώνει την μέγιστη κατανάλωση του συστήματος.

2.4 Μέτρηση δεδομένων από τους αισθητήρες.

Η μέτρηση των δεδομένων γίνεται όπως φαίνεται παρακάτω:

Με την χρήση του ποτενσιόμετρου:

```
1.   SensorPot = analogRead(15);
2.      SensorPot = map (SensorPot, 1020, 0, 0, 100);
```

Με την χρήση του αισθητήρα υπερήχων:

```
1.  digitalWrite(trigPin, LOW);
2.  delayMicroseconds(2);
3.  digitalWrite(trigPin, HIGH);
4.  delayMicroseconds(10);
5.  digitalWrite(trigPin, LOW);
6.  duration = pulseIn(echoPin, HIGH);
7.  duration = map(duration ,2389, 239, 0, 100);
```

Εδώ παρατηρείτε ο κώδικας για την μέτρηση με τα αισθητήρια και την λειτουργία map() για την αντιστοίχιση της μεταβλητής σε ποσοστό στάθμης.

2.5 Έλεγχος διαρροής και αποστολή sms.

Με τον τρόπο της σύγκρισης που αναφέρθηκε παραπάνω ελέγχεται εάν υπάρχει διαρροή. Εάν υπάρχει γίνεται αποστολή sms με προειδοποιητικό μήνυμα και προσθήκη μηνύματος στο site.

Έλεγχος με την μέτρηση του ποτενσιόμετρου:

```
1.  if ((PotMax - SensorPot)>10)
2.          {
3.        leakageStatus="<font color=red><h1>There is a leakage.</h1></font>";
4.
5.           if (sms.SendSMS("+306979532390","there is a leakage (Pot)."))
6.           {
7.             lcd.setCursor(0,3);
8.             lcd.print("Leakage (Pot).");
9.             for(int i = 0; i< 5; i++)
10.         {
```

```
            lcd.backlight();
            delay(250);
            lcd.noBacklight();
            delay(250);
            }
            lcd.backlight();
          }
```

Έλεγχος με την μέτρηση του αισθητήρα υπερήχων:

```
  if ((durationMax - duration)>10)
         {
           leakageStatus="<font color=red><h1>There is a leakage.</h1></font>";
           if (sms.SendSMS("+306979532390","there is a leakage (Ul)."))
         {
           lcd.setCursor(0,1);
           lcd.print("Leakage (Ul).");
           for(int i = 0; i< 5; i++)
          {
           lcd.backlight();
           delay(250);
           lcd.noBacklight();
           delay(250);
           }
           lcd.backlight();

          }
```

2.6 Έλεγχος νέων sms και απάντηση.

Η διαπίστωση εάν υπάρχει κάποιο εισερχόμενο sms γίνεται μέσω της εντολής `sms_position=sms.IsSMSPresent(SMS_UNREAD);` δηλαδή γίνεται έλεγχός της σημαίας `SMS_UNREAD`, και με την βοήθεια της εντολής `GetSMS()` είναι δυνατή η διαχείριση του sms.

```
   if(started)
   {
    sms_position=sms.IsSMSPresent(SMS_UNREAD);
    if (sms_position)
        {
         sms.GetSMS(sms_position, phone_number, sms_text, 100);
         if (sms_text[0] == '1')
             {
               sms.SendSMS("+306979532390", sms_answerPot);
               sms.SendSMS("+306979532390", sms_answerUl);
             }
          else
               sms.SendSMS("+306979532390", "Send the number 1, please.");

        }
```

3.Συμπεράσματα και μελλοντικές προσθήκες.

3.1 Συμπεράσματα.

Ο χρόνος εκπόνησης της εργασίας και η ενασχόληση με τον ελεγκτή στάθμης δεξαμενής πετρελαίου με διασύνδεση wifi και gsm, αποτέλεσε πηγή γνώσης και μάθησης, τόσο σε θεωρητικό επίπεδο των τεχνικών αλλά και της επεξεργασίας δεδομένων όσο και στο τεχνικό υπόβαθρο της ανάπτυξης της συγκεκριμένης μελέτης και υλοποίησης. Για τον μελετητή ήταν μία ευκαιρία για εμβάθυνση σε γνώσεις σχετικές με την επιστήμη της ηλεκτρονικής και πιο συγκεκριμένα με τα συστήματα επικοινωνιών και τα συστήματα αυτομάτου ελέγχου. Επίσης η ασχολία με την πλατφόρμα προγραμματισμού του Arduino βοήθησε στην κατανόηση της λειτουργίας των αλγορίθμων, των δομών προγραμματισμού, αλλά και των γλωσσών προγραμματισμού.

3.2 Μελλοντικές προσθήκες.

Οι μελλοντικές προσθήκες που θα μπορούσαν να γίνουν σε έναν τέτοιο πρωτότυπο ελεγκτή είναι οι ακόλουθες:

- Για αυτονομία, δηλαδή ανάπτυξη ενός συστήματος παράγωγης ενέργειας από ηλιακά πάνελ ή με μία αιολική γεννήτρια και αποθήκευση σε μπαταρίες για εξασφαλισμένη λειτουργία ακόμα και σε περιπτώσεις διακοπής ρεύματος ή για περιοχές που δεν έχουν πρόσβαση σε αυτό.
- Η δημιουργία μίας βάσεως δεδομένων για καταγραφή της κατανάλωσης θα βοηθούσε στην καλύτερη διαχείριση του υγρού καύσιμου.
- Ο προγραμματισμός του ESP8266 για αποστολή των δεδομένων μέσω του ίντερνετ οπουδήποτε.
- Διασύνδεση με σύστημα ασφαλείας με αισθητήρα καπνού για την δυνατότητα απευθείας κατάσβεσής κάποιας φωτιάς.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Arduino IDE Software. https://www.arduino.cc/en/Main/Software
2. Level Gauges Information http://www.globalspec.com/learnmore/sensors_transducers_detectors/level_sensing/level_gauges
3. C-Level Tank Level Gauge https://www.dandhgroup.co.uk/c-level-tank-gauge-level-indicator/
4. HC-SR04 Datasheet http://www.micropik.com/PDF/HCSR04.pdf
5. How I2C Communication Works and How To Use It with Arduino http://howtomechatronics.com/tutorials/arduino/how-i2c-communication-works-and-how-to-use-it-with-arduino/
6. Fritzing http://fritzing.org/home/
7. Everything ESP8266 http://www.esp8266.com/index.php
8. Kolban's Book on ESP [Neil Kolban] http://neilkolban.com/tech/esp8266/
9. GSM https://el.wikipedia.org/wiki/Global_System_for_Mobile_Communications
10. ESP 8266 Datasheet https://cdn-shop.adafruit.com/product-files/2471/0A-ESP8266__Datasheet__EN_v4.3.pdf
11. Arduino GSM library https://www.arduino.cc/en/Reference/GSM
12. Arduino MEGA 2560 REV 3 https://store.arduino.cc/arduino-mega-2560-rev3
13. Flowchart Maker https://www.draw.io/

ΠΑΡΑΡΤΗΜΑΤΑ

Κώδικας λειτουργίας προγράμματος.

```
1.  //Include Libraries
2.    //GSM
3.      #include "SIM900.h"
4.      #include <SoftwareSerial.h>
5.      #include "sms.h"
6.    //LCD
7.      #include <Wire.h>
8.      #include <LiquidCrystal_I2C.h>
9.
10. //Defines
11.   //ESP
12.     #define DEBUG true
13.   //Ultrasonic
14.     #define trigPin 53
15.     #define echoPin 52
16.
17.
18. //variables
19.   //ESP
20.     String leakageStatus;
21.   //GSM
22.     SMSGSM sms;
23.     int numdata;
24.     boolean started=false;
25.     char smsbuffer[160];
26.     char n[20];
27.     char sms_position;
28.     char phone_number[20];
29.     char sms_text[15];
30.     char sms_answerPot[100];
31.     String str1,str2,str3,str4;
32.     char sms_answerUl[100];
33.   //LCD
34.     // Set the pins on the I2C chip used for LCD connections:
35.     //                    addr, en,rw,rs,d4,d5,d6,d7,bl,blpol
36.     LiquidCrystal_I2C lcd(0x3F, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
37.   //Potensiometer
38.     int SensorPot = 0;
39.     int PotValue = 0 ;
40.     int PotMax = 0 ;
41.   //Ultrasonic
42.     long duration = 0;
43.     int durationMax = 0;
44.
45.
46. void setup(){
47.
48.   //LCD
49.     lcd.begin(20,4);
50.     lcd.clear();
51.     onoma();
52.     //GSM
53.     Serial1.begin(9600);
54.     if (gsm.begin(9600))
55.     {
56.       // ------- Quick 3 blinks of backlight  -------------
57.          for(int i = 0; i< 3; i++)
58.          {
59.            lcd.backlight();
```

```
            delay(250);
            lcd.noBacklight();
            delay(250);
            }
            lcd.backlight();
       ////////////////////////////////////////////////
       lcd.clear();
       lcd.setCursor(0,0);
       lcd.print("Status GSM = READY");
       started=true;
        }
       else
        {
         lcd.clear();
         lcd.setCursor(0,0);
         lcd.println("Status GSM = IDLE => Restart Module.");
          }
     if(started)
       {
        if (sms.SendSMS("+306979532390", "Sent 1 to find level."))
           {
             lcd.clear();
             lcd.setCursor(0,0);
             lcd.print("SMS has been Sent.");
           }
         else
           {
             lcd.setCursor(0,0);
             lcd.print("Error sending SMS.");
           }
         }

  //ESP
    leakageStatus ="";
    Serial2.begin(115200);
    sendData("AT+RST\r\n",2000,DEBUG); // reset module
    sendData("AT+CWMODE=2\r\n",1000,DEBUG); // configure as access point
    lcd.setCursor(0,2);
    lcd.print("IP : 192.168.4.1");
    sendData("AT+CIFSR\r\n",1000,DEBUG); // get ip address

            //lcd.print(response);
            sendData("AT+CIPMUX=1\r\n",1000,DEBUG); // configure for multiple conn
ections
            sendData("AT+CIPSERVER=1,80\r\n",1000,DEBUG); // turn on server on por
t 80

         //Potensiometer
           SensorPot = analogRead(15);
           SensorPot = map (SensorPot, 1020, 0, 0, 100);
           PotMax = SensorPot;

         //Ultrasonic
           pinMode(trigPin, OUTPUT);
           pinMode(echoPin, INPUT);
           digitalWrite(trigPin, LOW);
           delayMicroseconds(2);
           digitalWrite(trigPin, HIGH);
           delayMicroseconds(10);
           digitalWrite(trigPin, LOW);
           duration = pulseIn(echoPin, HIGH);
           duration =map(duration ,2389, 239, 0, 100);
           durationMax = duration;

       ///LCD
         onoma();
```

```
124.          delay(3000);
125.          lcd.clear();
126.        }
127.
128.
129.
130.        void loop()
131.           {
132.          //Potensiometer
133.            SensorPot = analogRead(15);
134.            SensorPot = map (SensorPot, 1020, 0, 0, 100);
135.
136.             //PotValue to char and construction of the answer_sms
137.               str1 = String(SensorPot);
138.               str2 = "Tank level (Potensiometer): " + str1 +" +-1.";
139.               str2.toCharArray(sms_answerPot,100);
140.
141.
142.          ///display to LCD
143.
144.              lcd.setCursor(0,2);
145.              lcd.print("Pot. value:         ");
146.              lcd.setCursor(0,2);
147.              lcd.print("Pot. value: "+str1);
148.               if ((PotMax - SensorPot)>10)
149.                 {
150.               leakageStatus="<font color=red><h1>There is a leakage.</h1></font>"
    ;
151.
152.                  if (sms.SendSMS("+306979532390","there is a leakage (Pot)."))
153.                  {
154.                    lcd.setCursor(0,3);
155.                    lcd.print("Leakage (Pot).");
156.                    for(int i = 0; i< 5; i++)
157.                {
158.                  lcd.backlight();
159.                  delay(250);
160.                  lcd.noBacklight();
161.                  delay(250);
162.                  }
163.                  lcd.backlight();
164.                 }
165.               }
166.
167.            PotMax = SensorPot ;
168.
169.            delay(2000);
170.         ///Ultrasonic//////////////
171.        pinMode(trigPin, OUTPUT);
172.        pinMode(echoPin, INPUT);
173.
174.        digitalWrite(trigPin, LOW);
175.        delayMicroseconds(2);
176.        digitalWrite(trigPin, HIGH);
177.        delayMicroseconds(10);
178.        digitalWrite(trigPin, LOW);
179.        duration = pulseIn(echoPin, HIGH);
180.        duration = map(duration ,2389, 239, 0, 100);
181.          /////distance to char and construction of the answer_sms
182.
183.                str3 = String(duration);
184.                str4 = "Tank level (Ultrasonic): " + str3 +" +-1.";
185.                str4.toCharArray(sms_answerUl,100);
186.
187.          ///display to LCD
188.
```

```
189.             lcd.setCursor(0,0);
190.             lcd.print ("Ult. value:        ");
191.             lcd.setCursor(0,0);
192.             lcd.print("Ult. value: "+str3);
193.
194.
195.        //Check for leakage
196.             if ((durationMax - duration)>10)
197.               {
198.                 leakageStatus="<font color=red><h1>There is a leakage.</h1></fon
    t>";
199.                 if (sms.SendSMS("+306979532390","there is a leakage (Ul)."))
200.               {
201.                 lcd.setCursor(0,1);
202.                 lcd.print("Leakage (Ul).");
203.                 for(int i = 0; i< 5; i++)
204.                {
205.                  lcd.backlight();
206.                  delay(250);
207.                  lcd.noBacklight();
208.                  delay(250);
209.                  }
210.                  lcd.backlight();
211.
212.                }
213.                }
214.       durationMax = duration;
215.
216.         //GSM
217.           if(started)
218.         {
219.          sms_position=sms.IsSMSPresent(SMS_UNREAD);
220.          if (sms_position)
221.              {
222.               sms.GetSMS(sms_position, phone_number, sms_text, 100);
223.               if (sms_text[0] == '1')
224.                   {
225.                     sms.SendSMS("+306979532390", sms_answerPot);
226.                     sms.SendSMS("+306979532390", sms_answerUl);
227.                   }
228.                else
229.                     sms.SendSMS("+306979532390", "Send the number 1, please.");

230.               }
231.             if (sms_position >= 5)
232.                {
233.                  for(int i=1;i<=5;i++)
234.                   {
235.                  sms.DeleteSMS(i);
236.                   }
237.                 }
238.           }
239.
240.
241.         //ESP
242.         if(Serial2.available()) // check if the esp is sending a message
243.         {
244.           if(Serial2.find("+IPD,"))
245.             {
246.              delay(1000);
247.
248.              int connectionId = Serial2.read()-
    48; // subtract 48 because the read() function returns
249.                                                  // the ASCII decimal value and
    0 (the first decimal number) starts at 48
250.
```

```
251.              String webpage = "<html><center><img src=http://www.autom.teithe.gr
    /img/logogr.png><h1>Ptyxiakh ergasia <p>Konstantinou Angelou</h1><p><h2>Potensiom
    eter : ";
252.                      webpage += SensorPot;
253.                      webpage += " +-2% .</h2><p><h2>Ultrasonic : ";
254.                      webpage += duration;
255.                      webpage += " +-2% .<br><br>";
256.                      webpage += leakageStatus;
257.              String cipSend = "AT+CIPSEND=";
258.              cipSend += connectionId;
259.              cipSend += ",";
260.              cipSend +=webpage.length();
261.              cipSend +="\r\n";
262.
263.              sendData(cipSend,1000,DEBUG);
264.              sendData(webpage,1000,DEBUG);
265.               webpage="<input type=\"button\"name=\"b1\"value=\"Refresh\"onclick
    =\"location.href='/'\"></center></html>";
266.
267.            cipSend = "AT+CIPSEND=";
268.            cipSend += connectionId;
269.            cipSend += ",";
270.            cipSend +=webpage.length();
271.            cipSend +="\r\n";
272.
273.            sendData(cipSend,1000,DEBUG);
274.            sendData(webpage,1000,DEBUG);
275.
276.
277.              String closeCommand = "AT+CIPCLOSE=";
278.              closeCommand+=connectionId; // append connection id
279.              closeCommand+="\r\n";
280.
281.              sendData(closeCommand,3000,DEBUG);
282.          }
283.         }
284.        }
285.
286.
287.       //LCD print name
288.       void onoma()
289.           {
290.             lcd.clear();
291.             lcd.setCursor(0,0);
292.             lcd.print("Ptyxiakh ergasia : ");
293.             lcd.setCursor(0,2);
294.             lcd.print("Konstantinou");
295.             lcd.setCursor(13,3);
296.             lcd.print("Angelou");
297.           }
298.
299.
300.       //ESP SendData function
301.          String sendData(String command, const int timeout, boolean debug)
302.         {
303.             String response = "";
304.
305.             Serial2.print(command); // send the read character to the esp8266
306.
307.             long int time = millis();
308.
309.             while( (time+timeout) > millis())
310.             {
311.               while(Serial2.available())
312.               {
313.
```

```
314.                 // The esp has data so display its output to the serial window
315.                 char c = Serial2.read(); // read the next character.
316.                 response+=c;
317.               }
318.             }
319.
320.             if(debug)
321.             {
322.
323.             }
324.
325.             return response;
326.         }
```