ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΤΜΗΜΑ ΒΙΟΜΗΧΑΝΙΚΗΣ ΣΧΕΔΙΑΣΗΣ ΚΑΙ ΠΑΡΑΓΩΓΗΣ

ΔΗΜΙΟΥΡΓΙΑ ΣΥΣΚΕΥΗΣ ΑΝΙΧΝΕΥΣΗΣ ΦΡΑΓΗΣ ΦΡΕΑΤΙΟΥ ΟΜΒΡΙΩΝ ΣΕ ΣΥΝΔΥΑΣΜΟ ΜΕ ΤΗΝ ΘΕΩΡΗΤΙΚΗ ΜΕΛΕΤΗ ΠΡΟΒΛΕΠΤΙΚΩΝ ΑΛΓΟΡΙΘΜΩΝ ΓΙΑ ΤΗΝ ΠΡΟΒΛΕΨΗ ΠΛΗΜΜΥΡΑΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Καμάρα Ελένη Δέσποινα

Επιβλέπων: Γρηγόριος Νικολάου, Καθηγητής ΠΑΔΑ

Αθήνα, Μάρτιος 2021

Μέλη εξεταστικής επιτροπής

1. Νικολάου Γρηγόριος
2. Βασιλειάδου Σουλτάνα
3. Δρόσος Χρήστος

GRIGORIOS NIKOLAOU Digitally signed by GRIGORIOS NIKOLAOU Date: 2021.03.09 11:42:41 +02'00'

Soultana Vasileiadou Digitally signed by Soultana Vasileiadou Date: 2021.03.16 11:03:26 +02'00'

Christos Drosos Digitally signed by Christos Drosos Date: 2021.03.16 14:34:02 +02'00'

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ/ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο/η κάτωθι υπογεγραμμένος/η Καμάρα Ελένη Δέσποινα του Σαντίγκιε, με αριθμό μητρώου 71445045 φοιτητής/τρια του Πανεπιστημίου Δυτικής Αττικής της Σχολής…….. του Τμήματος Μηχανικών Βιομηχανικής Σχεδίασης και Παραγωγής., δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της πτυχιακής/διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο/Η Δηλών/ούσα

Περίληψη

Η διπλωματική αυτή χωρίζεται σε δύο τομείς. Το πρώτο και βασικότερο κομμάτι αναφέρεται στην υλοποίηση μιας συσκευής, η οποία εισέρχεται εντός ενός φρεατίου ομβρίων υδάτων και έχει σαν σκοπό την ανίχνευση της φραγής του. Η συσκευή αποτελείται από ένα μικροελεγκτή Arduino UNO με ένα wi-fi module που, ουσιαστικά, το μετατραπεί σε IoT (Διαδίκτυο των Αντικειμένων) συσκευή και σε συνδυασμό με αισθητήρα επίπλευσης που ανιχνεύει την την φραγή του σωλήνα εντός του φρεατίου, αισθητήρα στάθμης νερού για την μέτρηση της στάθμης του νερού και αισθητήρα υγρασίας που μετρά τα επίπεδα υγρασίας εντός του φρεατίου, μπορεί να καθορίστει η κατάσταση ενός φρεατίου. Όλα τα δεδομένα που συλλέγονται από τους αισθητήρες αποθηκεύονται στο Cloud. Ακόμη, στήθηκε μια ιστοσελίδα, όπου ενημερώνεται με τα δεδομένα των αισθητηρίων.

Το δεύτερο κομμάτι της διπλωματικής αφορά την θεωρητική μελέτη διαφόρων αλγορίθμων μηχανικής μάθησης με σκοπό να επιλεχθεί ο κατάλληλος αλγόριθμος ώστε να επιτευχθεί η πρόβλεψη μιας πλημμύρας. Με τα δεδομένα που παρέχονται από τα αισθητήρια σε συνδυασμό με άλλα δεδομένα όπως τα γεωμορφολογικά στοιχεία μιας περιοχής, τα χιλιοστά και οι ώρες βροχής μπορεί να υλοποιηθεί ένας αλγόριθμος που θα επιστρέφει την πιθανότητα πλημμύρας ανάλογα με την κατάσταση των φρεατίων.

Αυτή η εργασία θα μπορούσε να βοηθήσει δήμους διότι θα ενημερώνονται real-time για την κατάσταση των φρεατίων και θα μπορούν να ενεργήσουν με ταχύτερους ρυθμούς σε περίπτωση ανάγκης. Επίσης, μπορεί να προσφέρει σημαντικές πληροφορίες που θα βοηθήσει στην ανακατασκευή ελαττωματικών φρεατίων.

Λέξεις κλειδια: Arduino UNO, Πλημμύρες, Διαδικτυο των Αντικειμένων, Μηχανική Μάθηση, HTML, PHP, AJAX,JS, MySQL

Abstract

This thesis is divided into two parts. The first part refers to the implementation of a device, which enters a rainwater well/manhole and aims to detect its blockage. The device consists of an Arduino UNO microcontroller with a wi-fi module, in order for it to be converted into an IoT (Internet of Things) device and in combination with a float sensor that detects the blockage of the pipe inside the well, a water level sensor to measure its level water and humidity sensor that measures the humidity levels inside the well, we can determine the condition of a rainwater well/manhole. All data collected by the sensors is stored in the Cloud. In addition, a website has been set up, where it is updated with the current data of the sensors.

The second part focuses on the theoretical study of various machine learning algorithms in order to achieve the prediction of a flood. With the data provided by the sensors in combination with other data such as the geomorphological data of an area, the millimeters and hours of rain, an algorithm can be implemented that will return the possibility of flooding depending on the condition of the wells.

This work could help municipalities because they will be informed in real-time about the condition of the wells and will be able to act faster in case of need. It can also provide important information that will help in the reconstruction of defective wells.

Key-words: Arduino UNO, Floods, Internet of Things, Machine Learning, HTML, PHP, AJAX,JS, MySQL

Ευχαριστίες

Στα πλαίσια της διπλωματικής μου εργασίας θα ήθελα να ευχαριστήσω τον υπεύθυνο καθηγητή για την εκπόνηση της εργασίας, κύριο Νικολάου Γρηγόρη, για όλη του την καθοδήγηση και βοήθεια. Επίσης, θα ήθελα να ευχαριστήσω τον συνάδελφο Σκιαδά Λυκούργο για την ανταλλαγή απόψεων και την συμβολή του στην υλοποίηση της εργασίας αυτής και τον αδερφό μου ,Γιάννη Καμάρα, για την βοήθεια που μου πρόσφερε στο κομμάτι του design της σελίδας.

Πίνακας Περιεχομένων

Κατάλογος Εικόνων

Κατάλογος Πινάκων

ΚΕΦΑΛΑΙΟ 1: ΕΙΣΑΓΩΓΗ

1.1 Εισαγωγή

Τα τελευταία χρόνια η ανθρωπότητα έχει έρθει αντιμέτωπη με αρκετές φυσικές καταστροφές, όπως σεισμούς, τσουνάμι, πλημμύρες πυρκαγιές κ.α. Αυτού του είδους οι φυσικές καταστροφές προκαλούν μεγάλες απώλειες, τόσο υλικές τόσο κι ανθρώπινες. Συνεπώς, ο άνθρωπος προσπαθεί με τεχνολογικά μέσα τα οποία έχει στην διάθεση του, να προβλέψει τις φυσικές καταστροφές πριν αυτές συμβούν, έτσι ώστε να μπορέσει να προετοιμαστεί κατάλληλα με σκοπό την διαφύλαξη της ανθρώπινης ζωής και έπειτα της περιουσίας που έχει χτίσει. Τέτοιο παράδειγμα βλέπουμε στις περιπτώσεις πυρκαγιών όπου σύμφωνα με στοιχεία της ΕΜΥ μπορούν να προβλεφθούν οι περιοχές στις οποίες υπάρχει πολύ υψηλός κίνδυνος πυρκαγιάς [1]. Αυτά τα τεχνολογικά μέσα, λοιπόν, δίνουν στους πολίτες τον κατάλληλο χρόνο να προετοιμαστούν και προφυλαχθούν από αυτά τα φαινόμενα.

Η παρούσα εργασία επικεντρώνεται στις πλημμύρες και πιο συγκεκριμένα με ποιους τρόπους θα μπορούσε να γίνει έγκαιρη πρόβλεψη και πρόληψη τους. Μια πλημμύρα είναι αρκετά δύσκολο να προβλεφθεί διότι υπάρχουν πάρα πολλοί παράγοντες που οδηγούν σε αυτήν, όπως η τοποθεσία της περιοχής, η γεωμορφολογία της, πόσες ώρες βρέχει και τα χιλιοστά νερού που θα πέσουν κατά την διάρκεια μιας βροχής. Ίσως, ο πιο σημαντικός παράγοντας, όμως, είναι τα φρεάτια ομβρίων υδάτων διότι συλλέγουν όλο το νερό της βροχής, συνεπώς σε περίπτωση δυσλειτουργίας τους (π.χ. σχάρες φραγμένες με σκουπίδια, σκουπίδια μέσα στο φρεάτιο, φραγή του σωλήνα εισροής νερού) αυξάνεται η πιθανότητα πλημμύρας στην περιοχή. Η λύση που προσφέρεται στην συγκεκριμένη διπλωματική στοχεύει πάνω στα φρεάτια ομβρίων υδάτων και αφορά μια συσκευή, που αποτελείται από διάφορους αισθητήρες και τοποθετείται εντός του φρεατίου δίνοντας την δυνατότητα ανίχνευσης της φραγής του φρεατίου μετά την πάροδο βροχής. Συνεπώς τοποθετώντας μία συσκευή σε κάθε φρεάτιο μιας περιοχής και έχοντας real-time ενημέρωση από αυτήν, θα μπορούσε να προβλεφθεί μια επερχόμενη πλημμύρα.

Πέρα από την ανίχνευση της φραγής του φρεατίου, η συσκευή με την χρήση των αισθητηρίων, θα προσφέρει πολλά χρήσιμα δεδομένα, όπως πόσος χρόνος απαιτείται για την φραγή ενός φρεατίου, πόσος χρόνος απαιτείται ώστε όλα τα φρεάτια μιας περιοχής φραγούν κτλ. Όλα αυτά τα δεδομένα μπορούν να βοηθήσουν ώστε να δημιουργηθεί ένας αλγόριθμος πρόβλεψης, ο οποίος θα παίρνει σαν είσοδους χαρακτηριστικά, όπως η γεωμορφολογία της περιοχής, την κατάσταση των φρεατίων, τα χιλιοστα βροχής, και θα προβλέπει την πιθανότητα πλημμύρας. Επίσης τα δεδομένα αυτά μπορούν να χρησιμοποιηθούν από δήμους έτσι ώστε να προβούν σε ανακατασκευή της αρχιτεκτονικης (όπως τις σωληνώσεις) των φρεατίων.

1.2. Smart Cities

Το smart city είναι ένας καινούργιος σχετικά όρος που έχει μπει στην καθημερινότητα. Τι ακριβώς είναι, όμως, το Smart City;

Σύμφωνα με τον ορισμό του McLaren, Duncan, Agyeman, Julian "Μία έξυπνη πόλη είναι μία αστική περιοχή που χρησιμοποιεί διαφορετικούς τύπους ηλεκτρονικών μεθόδων και αισθητηρίων για τη συλλογή δεδομένων. Οι γνώσεις που αποκτήθηκαν από αυτά τα δεδομένα χρησιμοποιούνται για τη βελτίωση των λειτουργιών σε όλη την πόλη" [2]

Με απλά λόγια, μια έξυπνη πόλη είναι μια πόλη που χρησιμοποιεί σύγχρονες τεχνολογίες (όπως το Διαδίκτυο των Αντικειμένων) και ηλεκτρονικά μέσα (όπως μικροελεγκτές και διάφορα αισθητήρια), για την επίλυση προβλημάτων της πόλης, σε τομείς όπως μεταφοράς/μετακίνησης, βελτίωσης κοινωνικών υπηρεσιών και βιωσιμότητας. Στην καθημερινότητα πλέον υπάρχουν τριγύρω μας διάφορα στοιχεία μιας έξυπνης πόλης, όπως για παράδειγμα:

1. Smart Parking, όπου μέσα από μια εφαρμογή στο κινητό ο πολίτης ενημερώνεται για διαθέσιμες θέσεις parking στην πόλη κι δίνεται η δυνατότητα κράτησης της θέσης.
2. Smart Lighting, όπου με την χρήση αισθητηρίων, τα οποία μετράνε το επίπεδο του φωτός ενεργοποιούνται τα φώτα της πόλης, σώζοντας έτσι ενέργεια.
3. Μόλυνση του αέρα. Με διάφορα αισθητήρια συλλέγονται οι μετρήσεις του αέρα για την παρακολούθηση της ποιότητας του σε συνεχή βάση.

Η συγκεκριμένη εργασία αποτελεί μια εφαρμογή που εμπεριέχεται στο "Smart City", διότι μπορεί να χρησιμοποιηθεί από δήμους με σκοπό να βελτιώσει τις λειτουργίες της πόλης.

1.3. Ιδέα

Αυτή η διπλωματική εργασία, όπως αναφέρθηκε στην εισαγωγή, διαπραγματεύεται τρόπους με τους οποίους μπορεί να γίνει πρόληψη κι πρόβλεψη πλημμυρών και πιο συγκεκριμένα τρόπους με τους οποίους μπορεί να γίνει έγκαιρη ενημέρωση στην περίπτωση φραγής ενός φρεατίου. Ουσιαστικά, η ιδέα μπορεί να χωριστεί σε δύο τομείς:

1. Κατασκευή συσκευής, η οποία θα τοποθετείται εντός του φρεατίου. Η συσκευή αυτή περιέχει αισθητήρες με τους οποίους μπορεί να καθορίστει το αν υπάρχει φραγή στην δίοδο που περνάει το νερό, το που έχει φτάσει η στάθμη του νερού αλλά κι γενικές πληροφορίες όπως η υγρασία κι η θερμοκρασία. Όλες αυτές οι πληροφορίες εμφανίζονται σε μια σελίδα ώστε να ενημερώνονται οι δήμοι για την κατάσταση όλων των φρεατίων μιας περιοχής.

2. Το 2ο σκέλος της διπλωματικής αυτής, έχει να κάνει με την θεωρητική μελέτη αλγορίθμων για την πρόβλεψη μιας πλημμύρας. Με την χρήση δεδομένων που μπορούν να συλλεχθούν από την συσκευή, άλλα κι άλλων χαρακτηριστικών, όπως η τοποθεσία και η γεωμορφολογία της περιοχής, τα χιλιοστά κι οι ώρες βροχής κ.α., δίνεται η δυνατότητα δημιουργίας ενός αλγόριθμου πρόβλεψης, ο οποίος μπορεί να προβλέψει κατά πόση πιθανότητα θα πλημμυρίσει. Συνεπώς, θα

δημιουργηθούν ορισμένα σενάρια κι συγκρίνοντας αλγόριθμους ταξινόμησης θα επιλεχθεί, θεωρητικά, ποιος αλγόριθμος είναι καταλληλότερος για αυτό το πρόβλημα.

1.4 Βιβλιογραφική Ανασκόπηση

Το Κεφάλαιο 2 ασχολείται με το θεωρητικό υπόβαθρο της εργασίας αυτής. Περιγράφονται περιληπτικά τα φρεάτια και το πως λειτουργούν. Επίσης περιγράφονται συνοπτικά όλες οι τεχνολογίες, ο μικροελεγκτής και τα αισθητήρια που χρησιμοποιήθηκαν για να χτίσουν την συσκευή και την ιστοσελίδα.

Το Κεφάλαιο 3 ασχολείται με τα στάδια κατασκευής της συσκευής και της ιστοσελίδας. Πιο συγκεκριμένα αναλύονται το σχηματικό κι οι ιδιαιτερότητες των αισθητηρίων αλλά κι η λογική που χρησιμοποιήθηκε στον κώδικα της συσκευής. Επιπροσθέτως, περιγράφεται αναλυτικά η λογική της ιστοσελίδας.

Το Κεφάλαιο 4 ασχολείται αποκλειστικά με το 2ο μέρος της ιδέας, που αφορά την πρόβλεψη μιας πλημμύρας. Αναλύονται, θεωρητικά, τα στάδια πρόβλεψης μιας πλημμύρας και συγκρίνοντας θεωρητικά κάποιους αλγόριθμους επιλέγεται το καταλληλότερο για το πρόβλημα που καλούμαστε να μελετήσουμε.

Το Κεφάλαιο 5 ασχολείται κυρίως με το πως θα συσκευαστεί και θα τοποθετηθεί η συσκευή εντός ενός φρεατίου. Αναφέρονται, επίσης, και μελλοντικές εξελίξεις της ιδέας.

Το Παράρτημα Α ασχολείται με τους πίνακες της βάσης δεδομένων, κάνοντας μια σύντομη περιγραφή τους.

Το Παράρτημα Β ασχολείται με τα APIs της σελίδας. Καταγράφονται όλα τα POST και GET APIs και η λειτουργία τους.

Το Παράρτημα Γ ασχολείται με την λειτουργία του κώδικα, αναλύοντας όλα τα AJAX κομμάτια του κώδικα.

ΚΕΦΑΛΑΙΟ 2: ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

2.1 Φρεάτια Ομβρίων Υδάτων

Προτού αναλυθούν οι τεχνολογίες που χρησιμοποιήθηκαν, πρώτα θα γίνει μια σύντομη περιγραφή των φρεατίων. Η συγκεκριμένη εργασία ασχολείται συγκεκριμένα με τα φρεάτια ομβρίων υδάτων τα οποία συναντώνται στους δρόμους και στα πεζοδρόμια, όπως φαίνεται στην εικόνα 2.1, και χρησιμοποιούνται για την υδροσυλλογή των νερών βροχής. Στον Δήμο Αθηναίων υπάρχουν περίπου 18000 φρεάτια [3].

Εικόνα 2.1: Εξωτερικό Φρεατίου (πηγή - ιστοσελίδα: https://www.stylianidis.gr/products/KalymmataFreation-Sxares/D-400)

Υπάρχουν πολλά είδη φρεατίων, κάποια πιο μακρόστενα, κάποια πιο στρογγυλά και κάποια που βρίσκονται στο πλάϊ των πεζοδρομίων. Σε αυτήν την διπλωματική, θα ασχοληθούμε με τα τετραγωνισμένα φρεάτια που έχουν την μορφή της διπλανής εικόνας. Παρακάτω παρουσιάζεται μία εικόνα με το εσωτερικό ενός τέτοιου φρεατίου. Όπως φαίνεται, το φρεάτιο έχει ένα σωλήνα(τρύπα), ο οποίος μπορεί είτε να βρίσκεται στο κάτω μέρος ή στη πλαϊνή πλευρά του φρεατίου. Από αυτόν τον σωλήνα, εισέρχεται το νερό της βροχής κι οδηγείται στο δίκτυο αποχέτευσης το οποίο έχει έναν κεντρικό σωλήνα για τα όμβρια ύδατα [4] .

Εικόνα 2.2: Εσωτερικό φρεατίου

Συνήθως τα φρεάτια καθαρίζονται μετά από προκηρύξεις πράγμα που σημαίνει ότι μπορεί να μην καθαρίζονται τακτικά. Κατά την διάρκεια βροχής, μπορεί να μεταφέρονται σκουπίδια τα οποία φράζουν τις σχάρες ή τον σωλήνα που βρίσκεται εντός του φρεατίου με αποτέλεσμα να βουλώνει. Στην περίπτωση που βουλώσει ένα φρεάτιο, τότε το επόμενο φρεάτιο θα δεχτεί μεγαλύτερη ποσότητα νερού με αποτέλεσμα να βουλώσει και αυτό κ.ο.κ.. Αν τα χιλιοστά βροχής είναι υψηλά και η περιοχή βρίσκεται κοντά σε ποτάμι ή βουνό τότε, προφανώς, αυξάνονται πάρα πολύ οι πιθανότητες πλημμύρας.

Αξίζει να σημειωθεί ότι πολλές φορές τα φρεάτια βουλώνουν κατά την διάρκεια μια μεγάλης βροχής λόγω του ότι συσσωρεύεται πολύ νερό στους σωλήνες, αλλά μετά το τέλος της βροχής το νερό ξεκινά να υποχωρεί. Σε αυτές τις περιπτώσεις, ορίζονται τα 30 λεπτά ως ο ελάχιστος χρόνος αναμονής μέχρι την υποχώρηση του νερού. Για να θεωρηθεί, λοιπόν, ένα φρεάτιο βουλωμένο σημαίνει ότι έχουν περάσει τα 30 λεπτά και τα νερά δεν έχουν υποχωρήσει.

Για την τοποθέτηση μιας συσκευής μέσα στο φρεάτιο, σίγουρα θα υπάρξουν δυσκολίες διότι πρέπει να τοποθετηθεί με τέτοιο τρόπο ώστε να μην εμποδίζει τον καθαρισμό των φρεατίων, η αλλαγή της μπαταρίας να γίνεται με γρήγορο τρόπο και να αντέχει στις αντιξοότητες, όπως καταιγίδες, μεγάλη ροή νερού εντός φρεατίου και χιόνια. Αυτό το κομμάτι θα αναλυθεί στο τελευταίο κεφάλαιο.

Τύποι Φρεατίων		
 Μακρόστενα Φρεάτια (Πηγή-ιστοσελίδα: https://www.makgratings.com/proionta/kanalia-freatia/)	 Φρεάτια στην άκρη του πεζοδρομίου (Πηγή-ιστοσελίδα: https://saint-gobain.gr/el/products/shares-selecta)	 Τετραγωνισμένα Φρεάτια (Πηγή-ιστοσελίδα: https://www.stylianidis.gr/products/KalymmataFreation-Sxares/D-400)

Πίνακας 2.1: Τύποι Φρεατίων

2.2 Τεχνολογίες

2.2.1 Διαδίκτυο των Πραγμάτων

Το Διαδίκτυο των Πραγμάτων ή IoT (Internet of Things) είναι ένας όρος που ακούγεται όλο και συχνότερα στην καθημερινή μας ζωής. Πολλές εταιρείες, κυρίως κινητης τηλεφωνίας, έχουν εισάγει αυτή την τεχνολογία στην Ελλάδα. Τι είναι όμως το Διαδίκτυο των Πραγμάτων;

“Το Διαδίκτυο των πραγμάτων ή Ίντερνετ των πραγμάτων (Internet of things ή IoT) αποτελεί το δίκτυο επικοινωνίας πληθώρας συσκευών, οικιακών συσκευών, αυτοκινήτων καθώς και κάθε αντικειμένου που ενσωματώνει ηλεκτρονικά μέσα, λογισμικό, αισθητήρες και συνδεσιμότητα σε δίκτυο ώστε να επιτρέπεται η σύνδεση και η ανταλλαγή δεδομένων.” [5]

Απλούστερα,

- Τα “πράγματα” αναφέρονται σε φυσικά στοιχεία, όπως για παράδειγμα οχήματα, οικιακές συσκευές, μοτέρ, μηχανές, τα οποία είναι ενσωματωμένα με λογισμικά, αισθητήρες ή/και ενεργοποιητές.

- Τα “πράγματα” είναι συνδεδεμένα , με τη χρήση του διαδικτύου, σε ένα cloud ή αλλιώς data center, το οποίο είναι ουσιαστικά ένας αποθηκευτικός χώρος για όλα τα δεδομένα που συλλέγονται από τους αισθητήρες.

- Τέλος τα δεδομένα που συλλέχθηκαν, επεξεργάζονται και απορρέουν διάφορα συμπεράσματα από αυτά.

Στην περίπτωση της διπλωματικής αυτής, το “πράγμα” είναι η συσκευή που θα τοποθετηθεί εντός του φρεατίου και θα είναι ενσωματωμένο με τρία αισθητήρια, φλοτέρ, υγρασίας κι στάθμης νερού. Η συσκευή συνδέεται στο διαδίκτυο και μεταφέρει όλα τα δεδομένα που συλλέγονται από τα αισθητήρια στο Cloud.

2.2.2 Μηχανική Μάθηση

Η μηχανική μάθηση είναι μια εφαρμογή τεχνητής νοημοσύνης (Artificial Intelligence ή AI). Επικεντρώνεται στην ανάπτυξη προγραμμάτων υπολογιστών που μπορούν να έχουν πρόσβαση σε δεδομένα και να τα χρησιμοποιούν για να μάθουν μόνα τους, χωρίς να χρειάζονται προγραμματισμό ή γενικότερα την ανθρώπινη παρουσία.

Το 1997 ο Tom M. Mitchell έδωσε πιο επίσημο ορισμό της μηχανικής μάθησης, ο οποίος χρησιμοποιείται ευρέως: "Ένα πρόγραμμα υπολογιστή λέγεται ότι μαθαίνει από μια εμπειρία E

σε σχέση μια σειρά από έργα T και μια μέτρηση της απόδοσης P, η οποία βελτιώνεται με την εμπειρία E".[6]

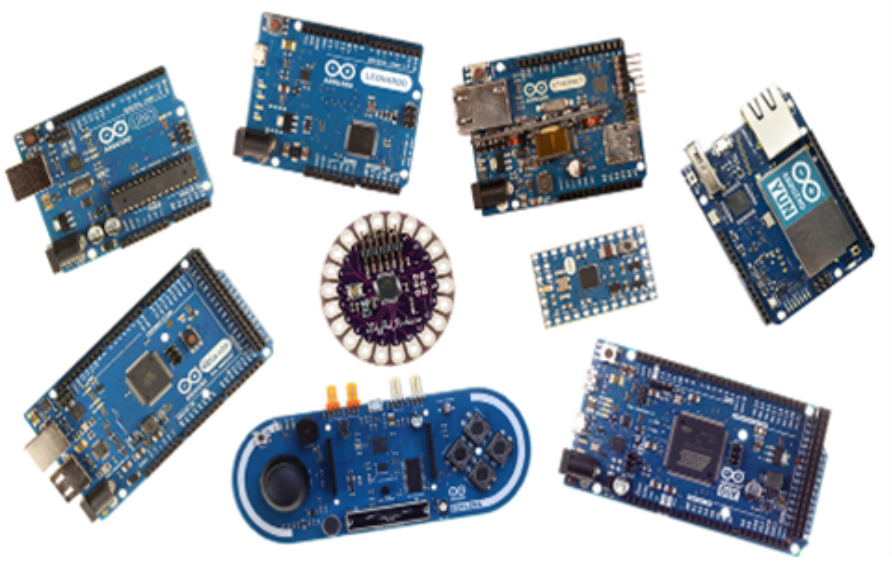
Όπως αναφέρθηκε στο προηγούμενο υποκεφάλαιο, μέσω της IoT τεχνολογίας συλλέγονται τα δεδομένα και αποθηκεύονται σε ένα cloud. Από αυτά τα δεδομένα μπορούν να εξαχθούν σημαντικές πληροφορίες και να αξιοποιηθούν. Οι πληροφορίες που εξάγονται από τα δεδομένα μπορεί να γίνει με την χρήση αλγορίθμων machine learning.

2.3 Προγραμματισμος συσκευής

Σε αυτό το υποκεφάλαιο αναλύονται ο μικροεπεξεργαστής και τα αισθητήρια τα οποία χρησιμοποιήθηκαν στην συσκευή.

2.3.1 Arduino

Το Arduino[7] είναι μια open source πλατφόρμα, η οποία μπορεί εύκολα να χρησιμοποιηθεί ώστε να κατασκευαστούν εφαρμογές ρομποτικής και συστήματα αυτοματισμού. Το Arduino αποτελείται από έναν ATmega μικροεπεξεργαστή της Atmel και από ψηφιακές και αναλογικές εισόδους/εξόδους. Πάνω σε αυτή την πλατφόρμα μπορούν να συνδεθούν διάφορα ηλεκτρονικά στοιχεία όπως αισθητήρες υγρασίας και θερμοκρασίας, αισθητήρες φωτός, διάφορες LCD οθόνες κτλ.



Εικόνα 2.3: Είδη Arduino (Πηγή- Ιστοσελίδα: https://www.elprocus.com/different-types-of-arduino-boards/)

Υπάρχει μια μεγάλη πληθώρα συσκευών Arduino,όπως φαίνεται στην παραπάνω εικόνα, που η κάθε μία έχει διαφορετικά προτερήματα. Για την συγκεκριμένη εργασία, επιλέχθηκε το Arduino Uno, διότι παρέχει 14 ψηφιακές εισόδους και 6 αναλογικές που είναι επαρκές για τον αριθμό των αισθητηρίων που θα χρησιμοποιηθούν[8]. Επίσης, ο μικροεπεξεργαστής ATmega328P είναι ικανοποιητικός αφού καταναλώνει αρκετή χαμηλή ενέργεια (15mA - κι ακόμη χαμηλότερα αν σκεφτούμε ότι η συσκευή θα είναι on sleep mode).

Εικόνα 2.4: Arduino UNO (Πηγή-Ιστοσελίδα: https://www.twinschip.com/Arduino-UNO-R3-SMD)

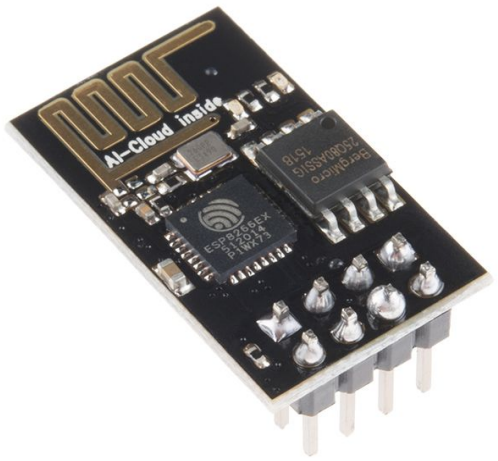
Σε περίπτωση που θέλαμε η συσκευή να έχει μικρότερο μέγεθος, τότε θα χρησιμοποιούσαμε ενα Arduino Nano που μας προσφέρει ότι ακριβώς και το UNO άλλα είναι υποδιπλάσιο του ή ακόμη καλύτερα, θα μπορούσε να δημιουργηθεί μια πλακέτα ειδικά σχεδιασμένη για το project αυτό.

Εικόνα 2.5: Arduino IDE

Για να προγραμματιστεί το Arduino, χρησιμοποιείται το Arduino IDE, το οποίο επιτρέπει στον χρήστη να γράψει C/C++ κώδικα και κι να "ανεβάσει" τον κώδικα στο Arduino. Ο κώδικας του Arduino αποτελείται από δύο standard συναρτήσεις, την setup() η οποία τρέχει μόνο μια φορά μόλις τεθεί η συσκευή σε λειτουργία και την loop(), η οποία τρέχει συνεχόμενα.

2.3.2 ESP8266-01

Το ESP8266 ESP-01 είναι ένα Wi-Fi module που επιτρέπει στους μικροελεγκτές να έχουν πρόσβαση σε ένα δίκτυο Wi-Fi. Χρησιμοποιώντας δηλαδή το ESP επιτυγχάνεται η σύνδεση του Arduino με το διαδίκτυο, μετατρέποντας το Arduino, τεχνικά, σε μια IoT συσκευή.



Εικόνα 2.6: ESP8266-01 Wifi Module(Πηγή - Ιστοσελίδα:
https://www.hellasdigital.gr/electronics/transceivers-and-communications/
esp8266-wifi-serial-transceiver-module-lwip-apsta-for-arduino/)

Για να επικοινωνήσει ένας client(π.χ. το Arduino Uno) με το ESP χρησιμοποιούνται τα AT Commands (Το AT είναι η συντομογραφία του ATtention). Οι εντολές AT είναι οδηγίες που χρησιμοποιούνται για τον έλεγχο ενός μόντεμ, GSM/GPRS modems και κινητών τηλεφώνων. Κάθε γραμμή εντολών ξεκινά με "AT" ή "at" [9]. Το αρχικό "AT" είναι το πρόθεμα που ενημερώνει το μόντεμ για την έναρξη μιας γραμμής εντολών.

Παρακάτω, παρατίθεται μια λίστα[10],[11] με τα πιο βασικά AT commands που χρησιμοποιούνται:

Command	Response	Περιγραφή
AT	OK	Είναι test command για να ελεγχθεί ότι δουλεύει σωστά η επικοινωνία του μικροελεγκτή με το ESP. Ο μικροελεγκτής στέλνει το AT και το ESP απαντάει με OK.
AT+CWMODE = WIFI mode (mode: 1, 2 or 3)	OK	Επιλογή του Wifi mode. Mode 1: Client mode (συνδεση σε modem) Mode 2: Host mode (esp λειτουργεί σαν

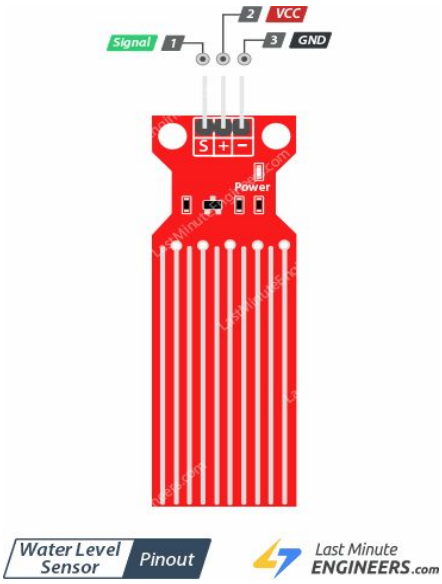
		hotspot) Mode 3: Host + Client mode
AT+CWJAP = ssid, pwd	OK	Εντολή ώστε να συνδεθεί το ESP, στο wifi network.
AT+CIPSTART=type,IP_address, port	OK	Εντολή ώστε να ξεκινήσει το connection, άρα να ξεκινήσει η μεταφορά των δεδομένων. Type: UDP ή TCP

Πίνακας 2.2: Βασικές AT εντολές

2.3.3 Αισθητήρες

2.3.2.1 Αισθητήρας σταθμής νερού

Το αισθητήριο στάθμης νερού αποτελείται από μια σειρά δέκα εκτεθειμένων ίχνων χαλκού, όπως φαίνεται στην παρακάτω εικόνα, πέντε από τα οποία είναι ίχνη ισχύος (στην εικόνα 2.7, τα ίχνη που έχουν το κυκλάκι) και πέντε είναι ίχνη αίσθησης. Αυτά τα ίχνη δεν είναι συνδεδεμένα αλλά γεφυρώνονται όταν βυθίζεται ο αισθητήρας στο νερό. Ουσιαστικά, η σειρά των ιχνών πάνω στο αισθητήριο, λειτουργεί ως μεταβλητή αντίσταση της οποίας η αντίσταση ποικίλλει ανάλογα με τη στάθμη του νερού.

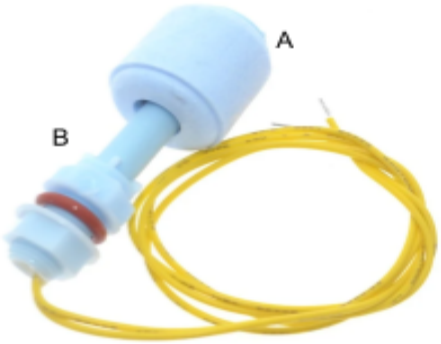


Εικόνα 2.7: Αισθητήριο στάθμης νερού (Πηγή - Ιστοσελίδα: https://lastminuteengineers.com/water-level-sensor-arduino-tutorial/)

Η αντίσταση είναι αντιστρόφως ανάλογη με το ύψος του νερού, δηλαδή σε όσο περισσότερο νερό βυθίζεται ο αισθητήρας, τόσο καλύτερη είναι αγωγιμότητα και θα έχει ως αποτέλεσμα χαμηλότερη αντίσταση. Αντιθέτως, όταν βυθίζεται λιγότερο νερό, τότε η αγωγιμότητα είναι κακή και θα συνεπάγεται σε υψηλότερη αντίσταση. Τέλος, ο αισθητήρας παράγει μια τάση εξόδου σύμφωνα με την αντίσταση, όπου με αυτή προσδιορίζεται η στάθμη του νερού.

2.3.2.2 Αισθητήρας επίπλευσης

Ένας αισθητήρας επίπλευσης ή αλλιώς φλοτέρ είναι μια συσκευή που χρησιμοποιείται για την ανίχνευση της στάθμης του υγρού μέσα σε μια δεξαμενή. Είναι ουσιαστικά ένας ηλεκτρομαγνητικός διακόπτης ON/OFF και αποτελείται από δύο επαφές σφραγισμένες σε γυάλινο σωλήνα. Όταν η επαφή με τον μαγνήτη πλησιάζει την άλλη επαφή, τότε έλκονται μεταξύ τους και αγγίζουν, επιτρέποντας στο ρεύμα να περάσει. Όταν ο μαγνήτης απομακρύνεται, οι επαφές απομαγνητίζονται και θα διαχωριστούν σπάζοντας το κύκλωμα. Ουσιαστικά με αυτό το αισθητήριο, θα ανιχνεύεται η φραγή του σωλήνα υδροσυλλογής μέσα στο φρεάτιο, δηλαδή όταν διακοπεί η εισροή του νερού τότε το αισθητήριο θα ενεργοποιείται.



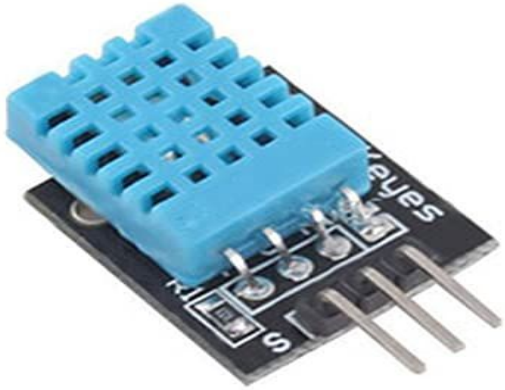
Εικόνα 2.8: Αισθητήριο επίπλευσης (Πηγή-Ιστοσελίδα: https://www.gadgetronicx.com/interfacing-float-sensor-arduino/)

Ο αισθητήρας που χρησιμοποιήθηκε είναι κανονικά ανοιχτός (NO: Normally Open), δηλαδή όταν ο πλωτήρας βρίσκεται στο χαμηλό του σημείο(σημείο A) όπως στην διπλανή εικόνα, τότε το κύκλωμα θα είναι ανοιχτό και όταν ο πλωτήρας βρίσκεται στο υψηλό του σημείο(σημείο B), θα κύκλωμα θα κλείσει.

2.3.2.3 Αισθητήριο υγρασίας - DHT11

Μέσα στο DHT11, υπάρχει ένα στοιχείο ανίχνευσης υγρασίας μαζί με ένα thermistor. Το στοιχείο ανίχνευσης υγρασίας έχει δύο ηλεκτρόδια με υπόστρωμα συγκράτησης υγρασίας μεταξύ τους. Τα ιόντα απελευθερώνονται από το υπόστρωμα καθώς οι υδρατμοί απορροφώνται από αυτό, γεγονός που με τη σειρά του αυξάνει την αγωγιμότητα μεταξύ των ηλεκτροδίων. Η

μεταβολή της αντίστασης μεταξύ των δύο ηλεκτροδίων είναι ανάλογη με τη σχετική υγρασία. Η υψηλότερη σχετική υγρασία μειώνει την αντίσταση μεταξύ των ηλεκτροδίων, ενώ η χαμηλότερη σχετική υγρασία αυξάνει την αντίσταση μεταξύ των ηλεκτροδίων. Έτσι λοιπόν, μετρώντας την αντίσταση προσδιορίζεται η υγρασία.



Εικόνα 2.9: Αισθητήριο υγρασίας DHT11 (Πηγή - Ιστοσελίδα: https://www.indiamart.com/proddetail/dht-11-temperature-humidity-sensor-module-13964138088.html)

Το DHT11, εκτός από την υγρασία, μετράει και την θερμοκρασία. Αυτό όμως δεν θα χρησιμοποιηθεί στα πλαίσια αυτής της διπλωματικής.

2.4 Προγραμματισμος Interface

Σε αυτό το υποκεφάλαιο αναλύονται οι τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση της ιστοσελίδας.

2.4.1 HTML5

HTML (HyperText Markup Language) είναι μια γλώσσα προγραμματισμού που χρησιμοποιείται για την σχεδίαση ιστοσελίδων. Αποτελείται από ειδικά σύμβολα που ονομάζονται tags (ετικέτες), τα οποία καθορίζουν την εμφάνιση του κειμένου, όπως παραγράφους ή συνδέσεις.

Στον παρακάτω πίνακα, απαριθμούνται κάποιες από τις πιο βασικές ετικέτες

Tag	Περιγραφή
<html> … </html>	Όλες οι ιστοσελίδες html ξεκινάνε με αυτήν την ετικέτα
<head> … </head>	Μέσα στην ετικέτα, περιέχονται πληροφορίες σχετικά με την ιστοσελίδα, όπως ο τίτλος της, το style που θα χρησιμοποιηθεί

	κτλ.
<title> … </title>	Περιέχει τον τίτλο της σελίδας
<body> … </body>	Αυτή η ετικέτα έρχεται αμέσως μετά από την head ετικέτα. Περιέχει όλα τα περιεχόμενα της σελίδας όπως κείμενα, εικόνες κτλ.
… ή/και h2, h3, h4, h5, and h6	Αυτή η ετικέτα χρησιμοποιείται για τον ορισμό επικεφαλίδων
…	Χρησιμοποιείται για την δημιουργία παραγράφων
<a> … 	Χρησιμοποιείται για την δημιουργία συνδέσμων
<img>	Χρησιμοποιείται για την εισαγωγή εικόνων στην ιστοσελίδα

Πίνακας 2.3: Βασικές HTML ετικέτες (Πηγή - Ιστοσελίδα: https://www.codebrainer.com/blog/top-10-html-tags)

Πρέπει να τονιστεί, ότι η HTML είναι μια στατική γλώσσα, που αυτό σημαίνει ότι οι σελίδες που δημιουργούνται δεν έχουν δυναμικό χαρακτήρα, δηλαδή περιέχει μόνο κείμενα και εικόνες, τα οποία δεν γίνεται να αλλάξουν το περιεχόμενο τους.

2.4.2 CSS3

Η CSS (Cascading Style Sheets) είναι μια γλώσσα που είναι άμεσα συνδεδεμένη με την HTML. Ουσιαστικά, ορίζει την στυλιστική εμφάνιση για τις σελίδες HTML. Στον παρακάτω πίνακα φαίνεται ένα παράδειγμα της χρήση της γλώσσα CSS (style="color:blue;") χρησιμοποιώντας την ετικέτα style μαζί με την html (<h1>A Blue Heading</h1>)

HTML και CSS	Οθόνη
A Blue Heading	Blue Heading

Πίνακας 2.4: Παράδειγμα HTML και CSS

2.4.3 PHP and SQL

Η PHP (Hypertext Preprocessor) είναι μια γλώσσα προγραμματισμού για τη δημιουργία ιστοσελίδων, αλλά διαφέρει από την HTML, διότι το περιεχόμενο μιας HTML σελίδας είναι στατικό, όπως ένα κείμενο. Αντιθέτως η PHP είναι δυναμική γλώσσα, που σημαίνει ότι το περιεχόμενο της σελίδας μπορεί να αλλάξει. Για παράδειγμα, μία μεταβλητή περιέχει έναν μετρητή που ξεκινά από το 0 και η τιμή της θα αλλάζει κάθε 1 λεπτό.

Μια άλλη διαφορά είναι ότι η HTML μπορεί να τρέξει κατευθείαν από το local PC, διότι μπορεί εύκολα να απεικονιστεί σε ενα web browser ενώ για την PHP, ο browser πρέπει να κάνει request στο web server ώστε να μπορέσει να απεικονίσει μια σελίδα .

Συχνά μαζί με την PHP χρησιμοποιείται η SQL. Η SQL είναι μια γλώσσα που χρησιμοποιείται στον προγραμματισμό για την διαχείριση δεδομένων ενός table στο database. Στον πίνακα 2.5, φαίνονται κάποιες από τις πιο βασικές εντολές της SQL.

Εντολή	Λειτουργία
SELECT * FROM table_name;	Η εντολή SELECT χρησιμοποιείται για την επιλογή δεδομένων από μια βάση δεδομένων.
UPDATE table_name SET column1 = value1, column2 = value2, ... WHERE condition;	Η εντολή UPDATE χρησιμοποιείται για την τροποποίηση των υπαρχουσών εγγραφών σε έναν πίνακα.
INSERT INTO table_name (column1, column2, column3, ...) VALUES (value1, value2, value3, ...);	Η εντολή INSERT INTO χρησιμοποιείται για την εισαγωγή νέων εγγραφών σε έναν πίνακα.
DELETE FROM table_name WHERE condition;	Η εντολή DELETE χρησιμοποιείται για την διαγραφή εγγραφών σε έναν πίνακα.

Πίνακας 2.5: Βασικές εντολές SQL

Πρέπει να σημειωθεί ότι η PHP και η SQL χρησιμοποιήθηκαν ώστε να δημιουργηθούν API. Τα API, ουσιαστικά, είναι μια εφαρμογή που έχει πρόσβαση στην βάση δεδομένων. Παρακάτω θα εξηγηθεί τι είναι το API με περισσότερες λεπτομέρειες,

2.4.4 JavaScript

Η JavaScript είναι γλώσσα που συνήθως χρησιμοποιείται να προσθέσει δυναμικό χαρακτήρα στην HTML (Δυναμική HTML).

Είναι δυναμική γλώσσα τόσο στη σύνταξή της, π.χ: μια μεταβλητή που περιέχει έναν αριθμό μπορεί αργότερα να αποθηκεύσει κείμενο, όσο και στην εκτέλεσή της που είναι runtime (το runtime παρέχει κανόνες όπως πώς μπορεί το πρόγραμμα να αλληλεπιδράσει με το λειτουργικό σύστημα του υπολογιστή και αν η σύνταξη προγράμματος είναι νόμιμη).

2.4.5 HTTP Requests και API

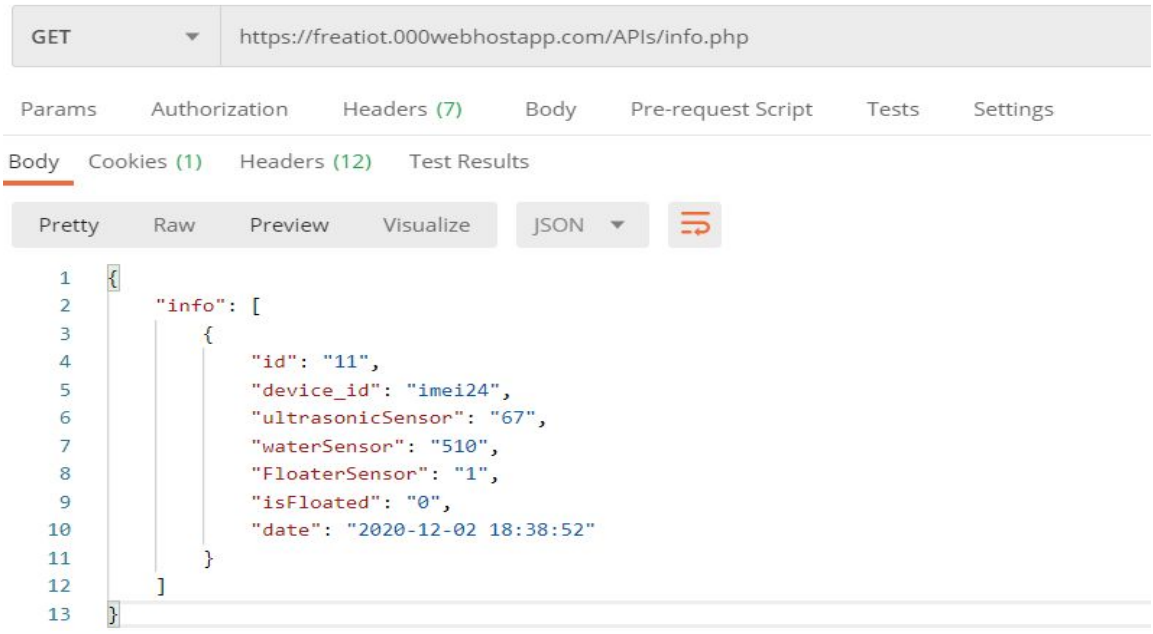
Στο κεφαλαιο 2.4.3 αναφέρθηκε ότι η PHP και η SQL χρησιμοποιήθηκαν για την δημιουργία των API της ιστοσελίδας. Το API (Application Programming Interface) έχει έναν αρκετά περίπλοκο ορισμό όποτε θα εξηγηθεί με όσο πιο απλά λόγια γίνεται. Δημιουργείται με τις γλώσσες SQL και PHP και έχει την μορφή ενός URL, όπως φαίνεται στον πίνακα 2.6.
Ουσιαστικά, αυτό που γίνεται στο API, είναι ότι μέσω της γλώσσας PHP επιτυγχάνεται σύνδεση στην βάση δεδομένων και χρησιμοποιώντας εντολές της SQL όπως για παράδειγμα "SELECT * FROM table", αποθηκεύονται τα δεδομένα που συλλέχθηκαν από την εντολή της sql στο URL. Ο λόγος που χρησιμοποιείται το API είναι γιατί μας επιτρέπει να δείξουμε αυτά τα δεδομένα με εύκολο τρόπο στην σελίδα.

Για να μπορέσουν να αξιοποιηθούν όλες τις χρήσεις ενός API και για να μπορέσει να ενσωματωθεί στην ιστοσελίδα, χρειαζόμαστε τα HTTP Requests. Τα HTTP Requests λειτουργούν ως ενδιάμεσος τρόπος μεταφοράς πληροφοριών μεταξύ ενός client ή εφαρμογής και ενός server. Ο client υποβάλλει ένα HTTP request στο server και ο server στέλνει μια απάντηση. Η απάντηση περιέχει τις πληροφορίες του API .
Υπάρχουν διάφοροι μέθοδοι HTTP requests, όπου η κάθε μέθοδος έχει έναν συγκεκριμένο σκοπό. Στον παρακάτω πίνακα παρουσιάζονται οι 2 πιο διαδεδομενοι μέθοδοι.

Request	Λειτουργία
GET https://freatiot.000webhostapp.com/APIs/info.php	Το GET χρησιμοποιείται για να ζητήσει δεδομένα από έναν server.
POST https://freatiot.000webhostapp.com/APIs/writedata.php?sensor=x1&sensor2=x2&created=date	Το POST χρησιμοποιείται για την αποστολή δεδομένων σε έναν server για τη δημιουργία νέας στήλης ή ενημέρωσης της βάσης. Τα δεδομένα που θα εισαχθούν βρίσκονται στο URL,δηλαδή π.χ. στην βάση η στήλη sensor παίρνει την τιμή x1, η στήλη sensor 2 παίρνει την τιμή x2 και η στήλη created παίρνει την τιμή του date.

Πίνακας 2.6: HTTP GET και POST Requests

Παρακάτω φαίνεται το αποτέλεσμα ενός GET HTTP Request με την χρήση του προγράμματος Postman[12].



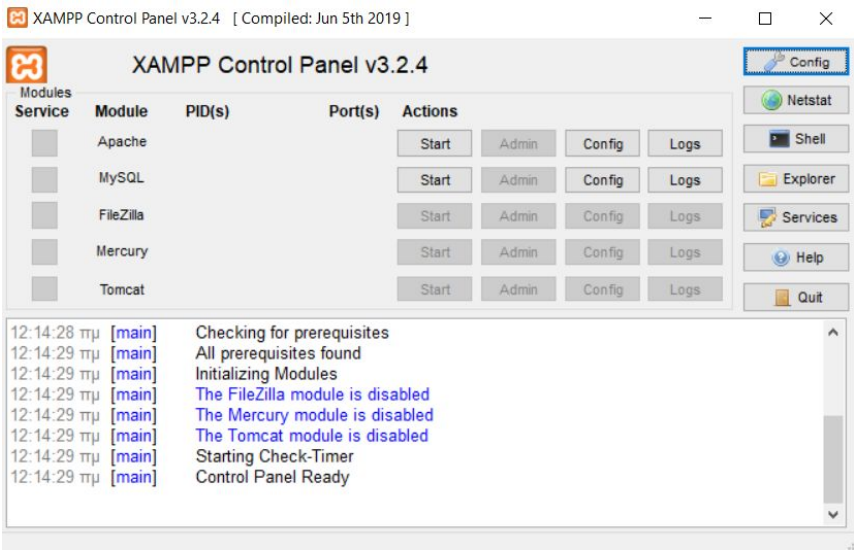
Εικόνα 2.10: Αποτέλεσμα ενός HTTP GET Request (πρόγραμμα: Postman)

2.4.6. Ajax

Η Ajax (Asynchronous JavaScript and XML) είναι ένα σύνολο από web development τεχνικές που χρησιμοποιούν πολλές client-side τεχνολογίες, όπως η JavaScript, για να δημιουργηθούν ασύγχρονες web εφαρμογές. Με Ajax, οι web εφαρμογές μπορούν να στέλνουν και να ανακτούν δεδομένα από έναν server ασύγχρονα (τρέχοντας στο παρασκήνιο), επιτρέποντας σε ιστοσελίδες να αλλάζουν το περιεχόμενο τους δυναμικά, χωρίς να χρειάζεται να φορτωθεί εκ νέου ολόκληρη η σελίδα. Στο Ajax κομμάτι, εφαρμόζονται τα HTTP Request ώστε τα δεδομένα που εξάγονται από το API να χρησιμοποιηθούν στην σελίδα και να ανανεώνονται αυτόματα κάθε φορά που υπάρχει νέα μετρηση, χωρίς όμως να πρέπει να φορτωθεί όλη η σελίδα.

2.3.7. Xampp

Το xampp είναι ένα πρόγραμμα που περιέχει έναν server Apache, βάση δεδομένων MySQL και ένα διερμηνέα για κώδικες γραμμένους σε γλώσσες προγραμματισμού PHP.

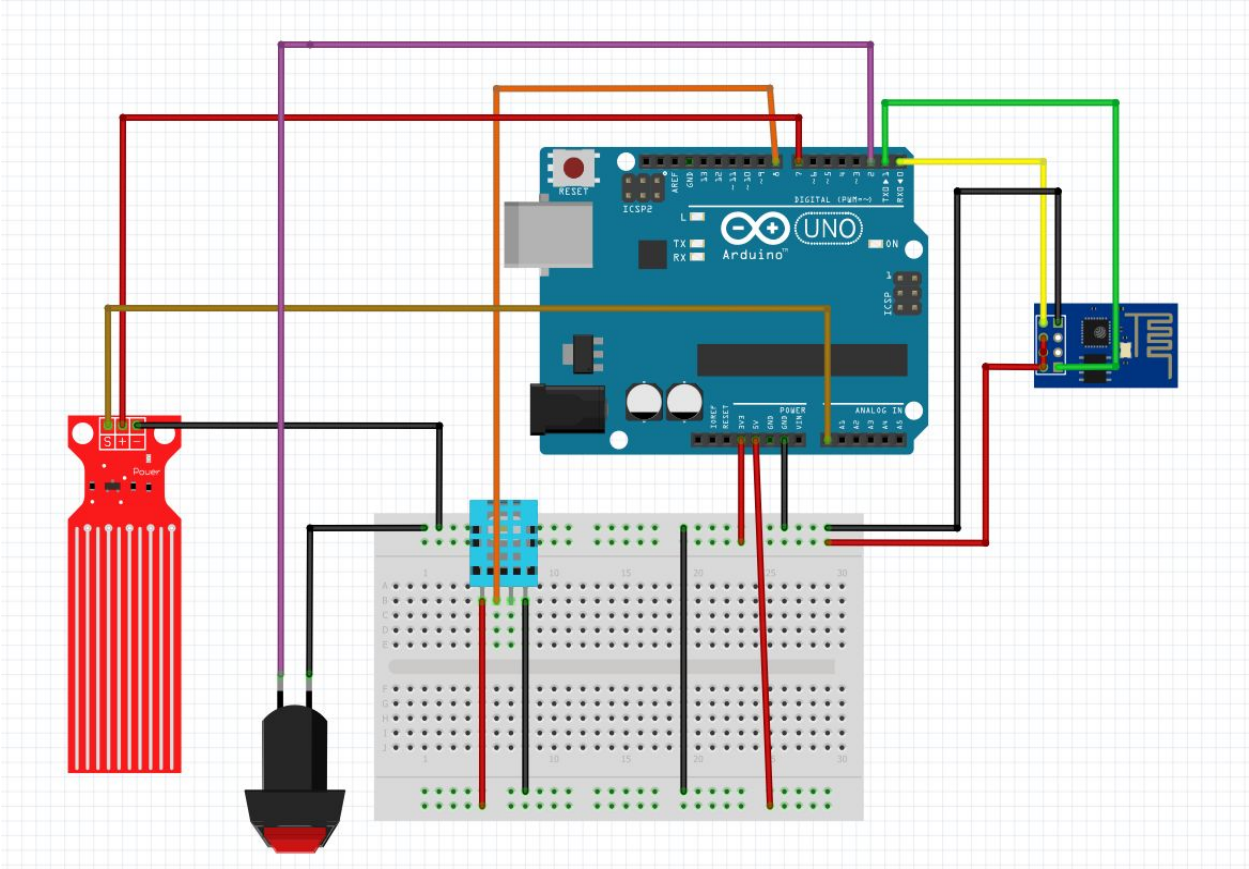


Εικόνα 2.11: Xampp (πρόγραμμα: Xampp Control Panel v3.2.4)

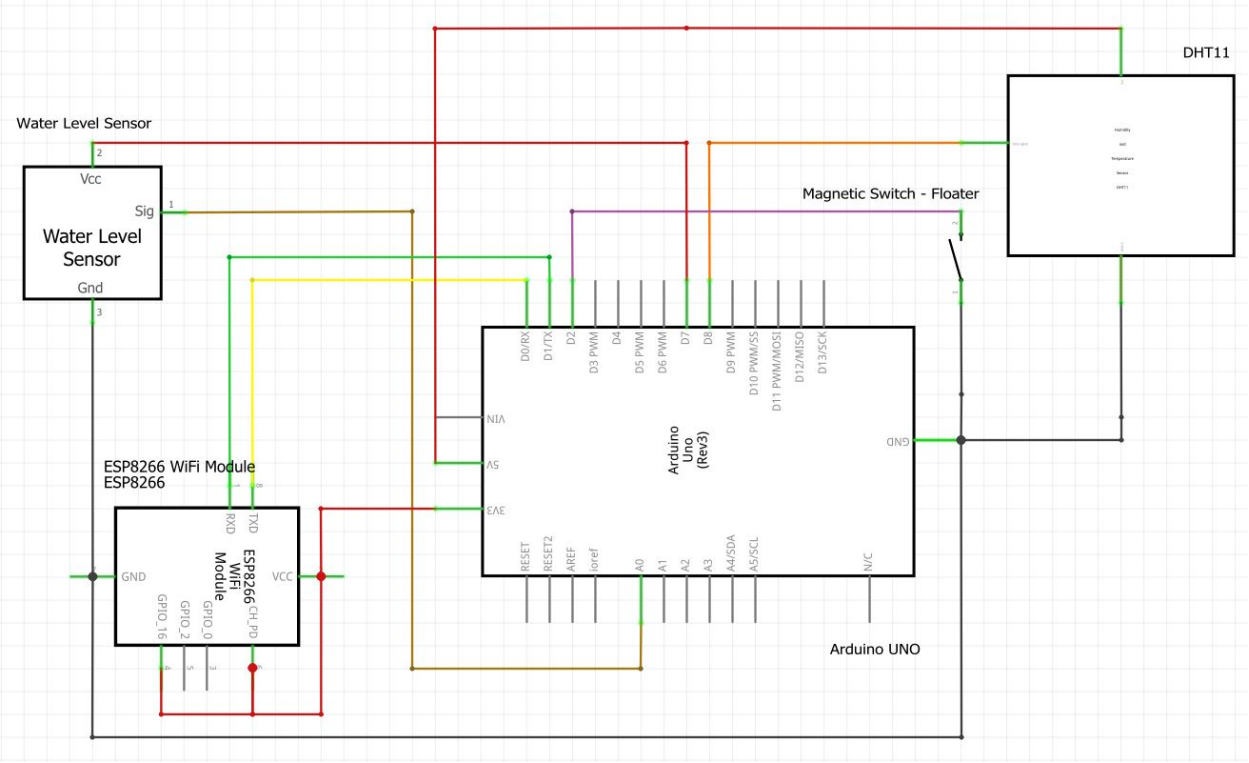
ΚΕΦΑΛΑΙΟ 3: ΚΑΤΑΣΚΕΥΗ ΤΗΣ ΙΔΕΑΣ

3.1 Σχηματικό

Παρακάτω, απεικονίζεται το κύκλωμα που είναι ‘χτισμένο’ πάνω στο breadboard αλλά και το σχηματικό. Θα αναλύσουμε το κάθε στοιχείο ξεχωριστά και τις ιδιαιτερότητες τους στα παρακάτω υποκεφάλαια. Ο σχεδιασμός των κυκλωμάτων έγινε με το πρόγραμμα Fritzing. [13]



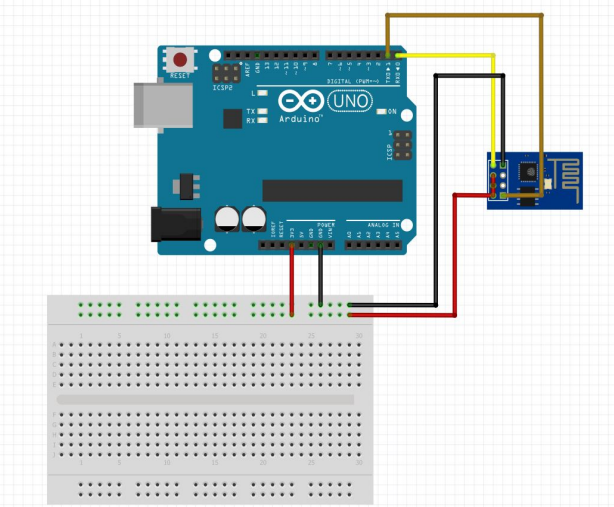
Εικόνα 3.1: Κύκλωμα Breadboard (πρόγραμμα: Fritzing v0.9.3)



Εικόνα 3.2: Σχηματικό (πρόγραμμα: Fritzing v0.9.3)

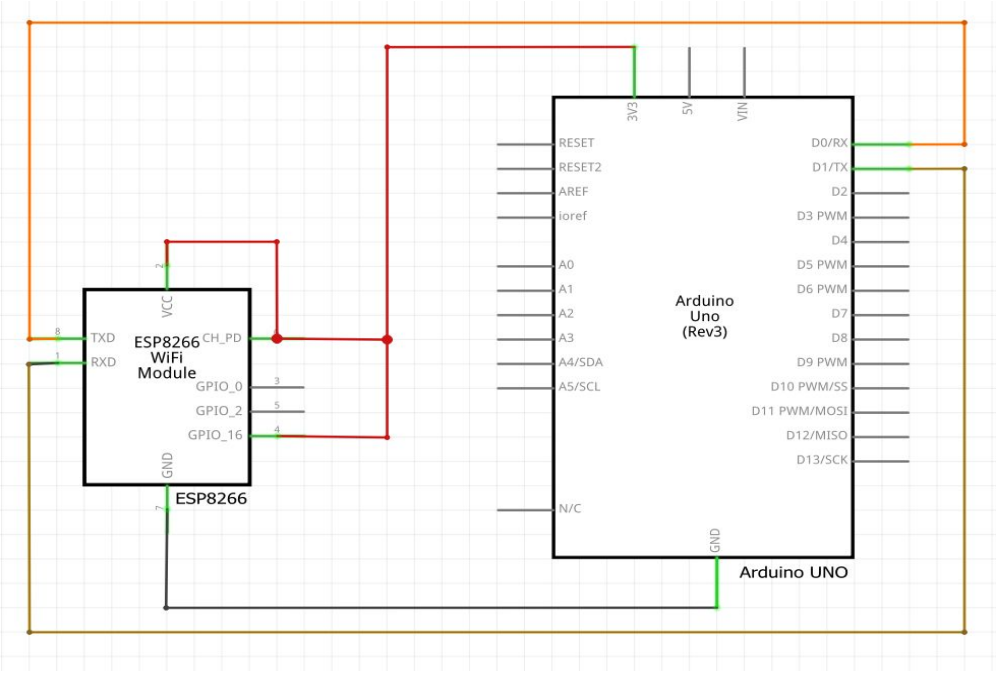
3.1.1. ESP8266-01 Wifi Module

Το ESP8266-01 [14] είναι η συσκευή που μετατρέπει το Arduino UNO από έναν απλό μικροελεγκτή σε μια συσκευή IoT. Μέσω του ESP επιτυγχάνεται η σύνδεση του Arduino με το internet.



Εικόνα 3.3: Κύκλωμα Breadboard του ESP8266-01 Wifi Module (πρόγραμμα: Fritzing v0.9.3)

Όπως φαίνεται στο παρακάτω κύκλωμα, τα pins VCC, CH_PD και RST(GPIO_16) συνδέονται στο 3.3volt. Το VCC είναι το pin της τάσης ενώ το CH_PD (Chip Power Down) και το RST (Reset) χρησιμοποιείται για να μπει το ESP σε sleep mode. Λόγω του ότι θέλουμε να το αποφύγουμε αυτό, το συνδέσαμε στα 3.3volt.
Η γείωση του ESP συνδέεται με το Arduino UNO. Τέλος, το RX του ESP συνδέεται με το TX του Arduino και το TX του ESP συνδέεται με το RX του Arduino, αντίστοιχα.

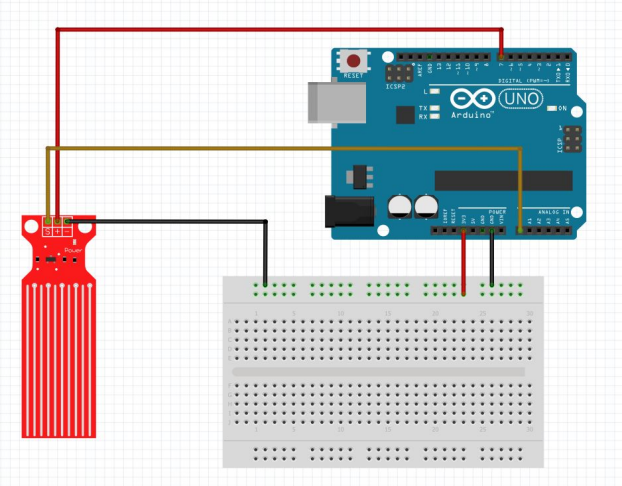


Εικόνα 3.4: Σχηματικό του ESP8266-01 Wifi Module (πρόγραμμα: Fritzing v0.9.3)

3.1.2 Αισθητήρας στάθμης νερού

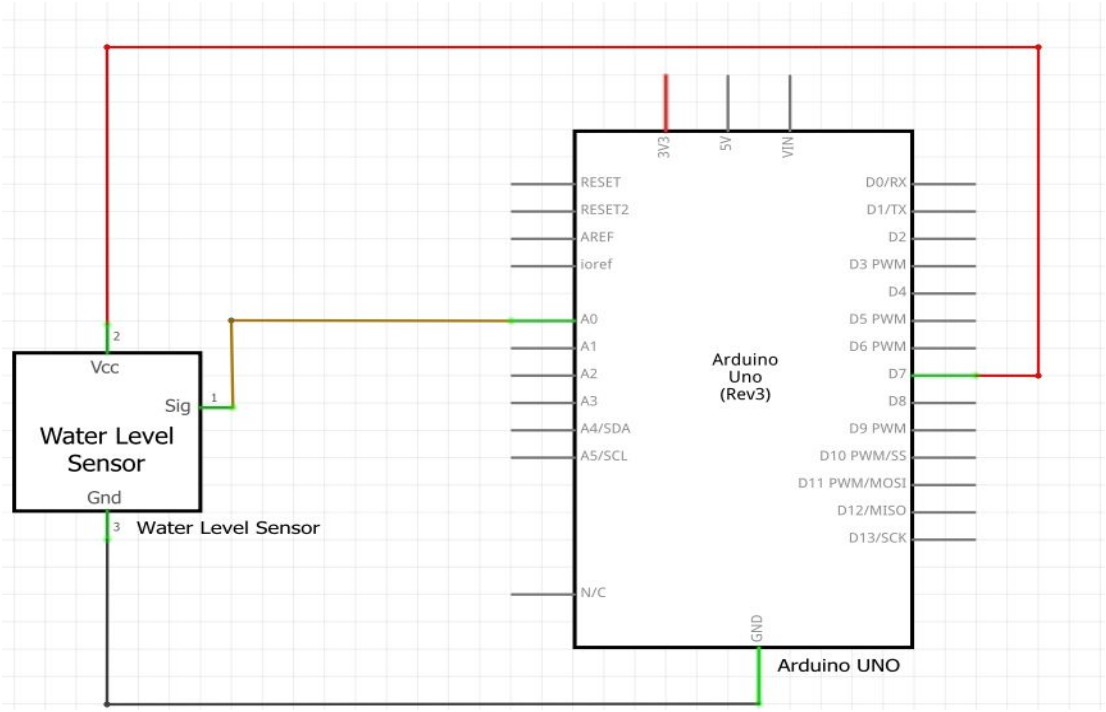
Ο αισθητήρας στάθμης νερού είναι ένα αναλογικό αισθητήριο, για αυτό τον λόγο συνδέεται στην αναλογική είσοδο A0 του Arduino. Οι τιμές που επιστρέφει είναι ανάμεσα:

- Στον αριθμό ‘0’ όταν δεν υπάρχει καθόλου νερό και
- Στον αριθμό ‘533’ όταν η στάθμη του νερού έχει φτάσει στο μέγιστο



Εικόνα 3.5: Κύκλωμα Breadboard του αισθητήρα στάθμης νερού (πρόγραμμα: Fritzing v0.9.3)

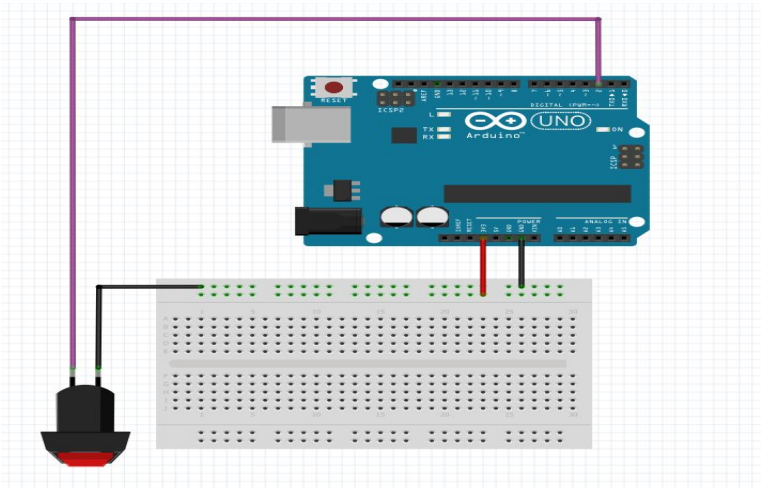
Επίσης, πρέπει να σημειωθεί ότι ο αισθητήρας αυτός είναι αρκετά ευαίσθητος στις υψηλές τάσεις. Συνεπώς, αντί να συνδεθεί κατευθείαν στα 5 Volt, ήταν προτιμότερο να συνδεθεί στην ψηφιακή είσοδο D7, η οποία ενεργοποιείται μόνο όταν χρειάζεται να πάρει μέτρηση ο αισθητήρας. Με αυτόν το τρόπο, ο αισθητήρας έχει μεγαλύτερο προσδόκιμο ζωής.



Εικόνα 3.6: Σχηματικό του αισθητήρα στάθμης νερού (πρόγραμμα: Fritzing v0.9.3)

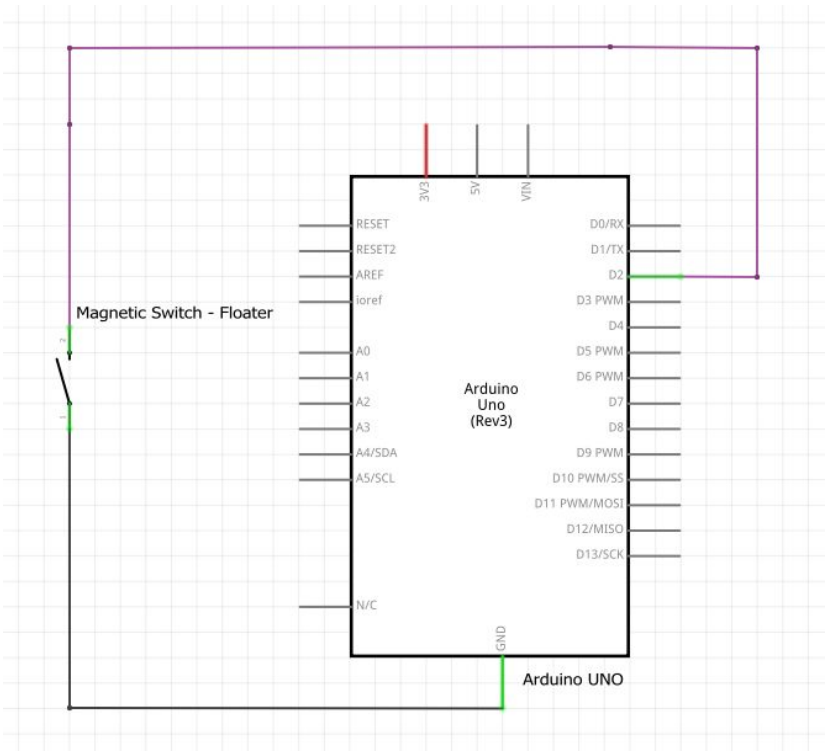
3.1.3. Αισθητήρας επίπλευσης

Ο αισθητήρα επίπλευσης, ουσιαστικά, λειτουργεί σαν διακόπτης, δηλαδή παίρνει δύο διακριτές τιμές, το 0 και το 1. Η συνδεσμολογία του είναι αρκετά απλή, όπως φαίνεται στην παρακάτω εικόνα.



Εικόνα 3.7: Κύκλωμα Breadboard του αισθητήριο επίπλευσης (πρόγραμμα: Fritzing v0.9.3)

Το ένα καλώδιο συνδέεται στην γείωση και το άλλο στην ψηφιακή είσοδο D2.

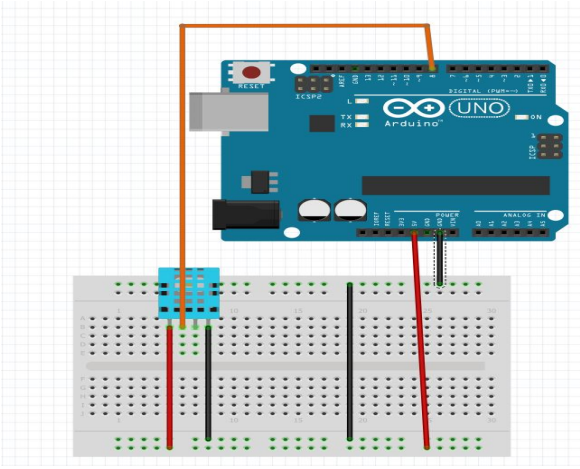


Εικόνα 3.8:Σχηματικό του αισθητήριο επίπλευσης (πρόγραμμα: Fritzing v0.9.3)

Όπως αναφέρθηκε στο κεφάλαιο 2.3.2.2, ο αισθητήρας είναι κανονικά ανοιχτός (NO: Normally Open), δηλαδή όταν ο πλωτήρας βρίσκεται στο χαμηλό του σημείο το κύκλωμα θα είναι ανοιχτό(τιμή 0) και όταν ο πλωτήρας βρίσκεται στο υψηλό του σημείο, θα ολοκληρώνει το κύκλωμα (τιμή 1).

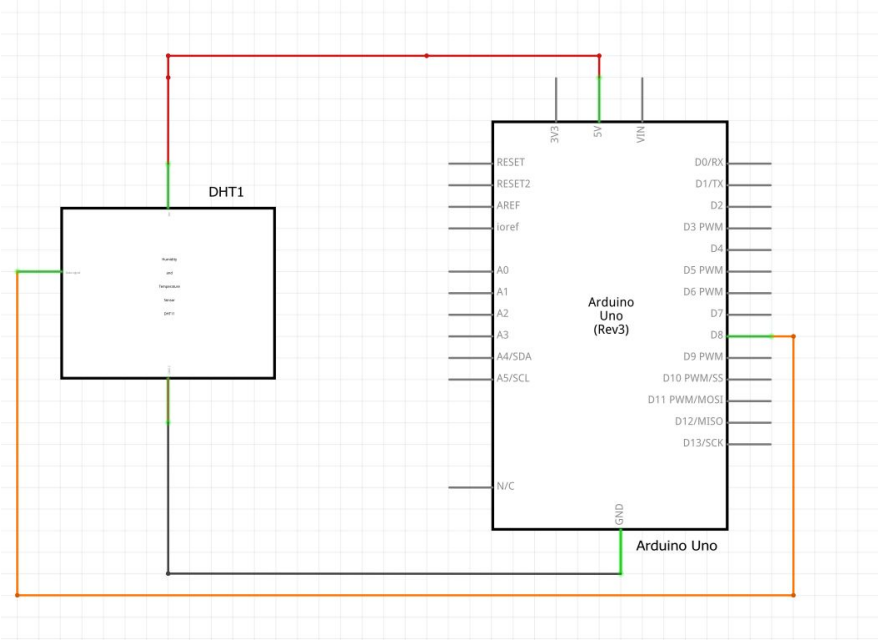
3.1.4. Αισθητήρας υγρασίας - DHT11

Το DHT11 έχει τρία ποδαράκια, το VCC (5V), την γείωση και το data signal το οποίο συνδέθηκε με την ψηφιακή είσοδο D8..



Εικόνα 3.9: Κύκλωμα Breadboard του αισθητήριο υγρασίας (πρόγραμμα: Fritzing v0.9.3)

Το αισθητήριο αυτό μετράει υγρασία και θερμοκρασία, αλλά στα πλαίσια της διπλωματικής αυτής χρησιμοποιήθηκε μόνο για την μέτρηση της υγρασίας.



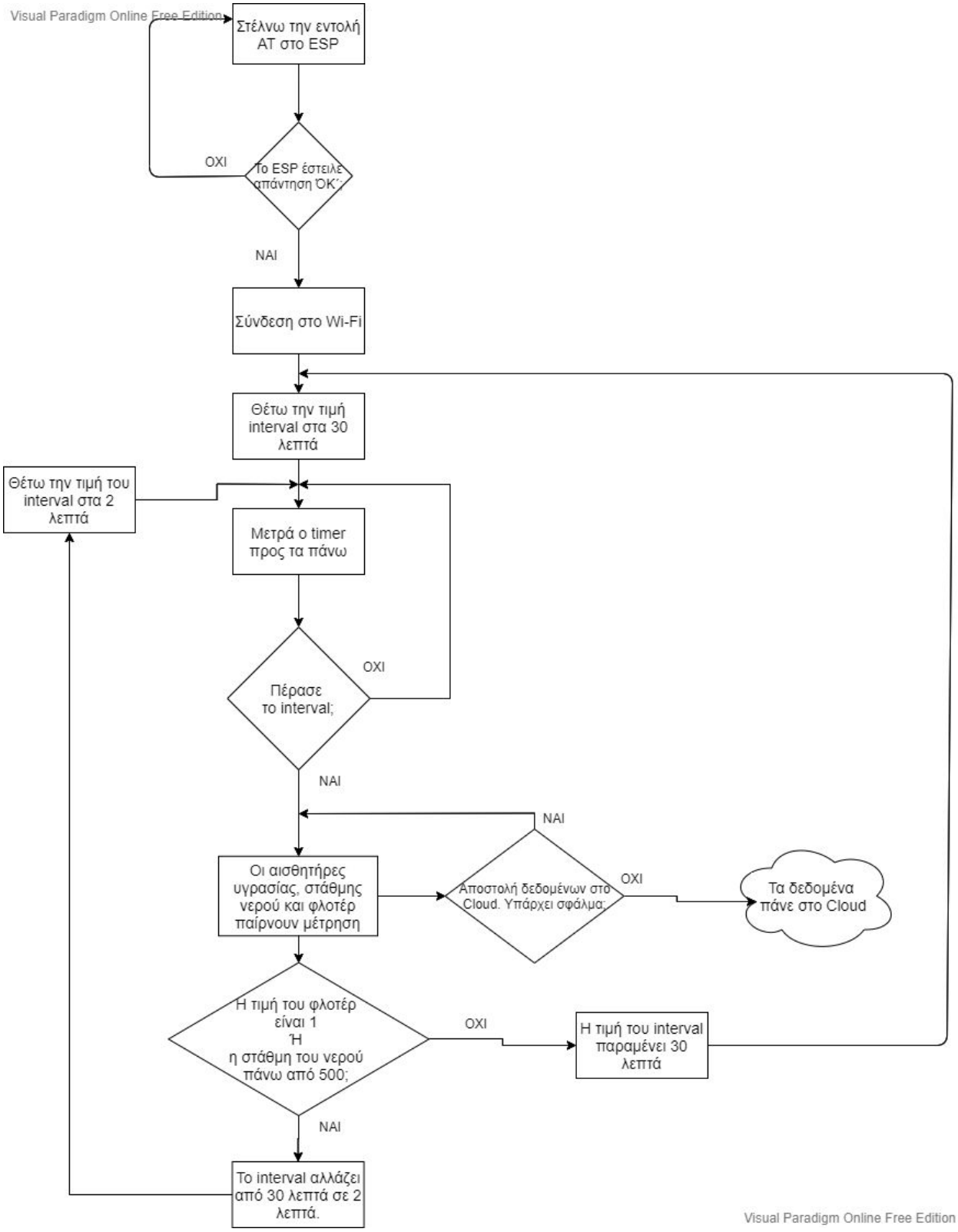
Εικόνα 3.10: Σχηματικό του αισθητήριο υγρασίας (πρόγραμμα: Fritzing v0.9.3)

3.2 Κώδικας συσκευής

Προτού αναλύσουμε τα επιμέρους στοιχεία του κώδικα, θα επισυναπτεί το λογικό διάγραμμα (flowchart) που εξηγεί την λογική της συσκευής. Η λογική που ακολουθήθηκε είναι αρκετά απλή. Αρχικά, στέλνεται ένα AT test command στο ESP και το ESP, με την σειρά του, πρέπει να στείλει την απάντηση “OK” . Μόλις ληφθεί το “OK”, σημαίνει ότι το ESP λειτουργεί και επικοινωνεί σωστά με το Arduino. Στην συνέχεια, συνδέεται το ESP με το WiFi και ξεκινά η διαδικασία της μέτρησης.

Η μέτρηση πραγματοποιείται κάθε 30 λεπτά και περιλαμβάνει τις μετρήσεις των αισθητηρίων υγρασίας, στάθμης νερού και του φλοτέρ. Σε περίπτωση που το φλοτέρ είναι ενεργοποιημένο (έχει την τιμή είναι 1) ή η στάθμη του νερού είναι πάνω από 500, τότε οι μετρήσεις πραγματοποιούνται κάθε 2 λεπτά, αντί για 30 λεπτά. Μετά από κάθε μέτρηση, εφόσον δεν υπάρξει κάποιο σφάλμα, τα δεδομένα αποθηκεύονται στο cloud.
Ο σχεδιασμός του flowchart έγινε με την χρήση του προγράμματος Virtual Paradigm Online Free Edition [15].

Στην συνέχεια του υποκεφαλαίου, θα αναλυθουν περιγραφικά ο κώδικας κι η λειτουργία της συσκευής.



Εικόνα 3.11: Λογικό διάγραμμα της συσκευής (πρόγραμμα: Virtual Paradigm Online Free Edition)

3.2.1. Επεξήγηση κώδικα

1. Βιβλιοθήκες

```
#include <stdlib.h>
#include <dht.h>
```

Οι βιβλιοθήκες που χρησιμοποιήθηκαν είναι οι:

- Stdlib , η οποία μας δίνει την δυνατότητα να χρησιμοποιήσουμε την συνάρτηση millis() ώστε να ορίσουμε κάθε πόσα λεπτά θα παίρνει μέτρηση η συσκευή
- Dht, η οποία μας προσφέρει τις συναρτήσεις “DHT.temperature” και “DHT.humidity” ώστε να πάρουμε μέτρηση από τον αισθητήρα υγρασίας DHT11

2. Αρχικοποιήσεις

```
//Variables for DHT
float temp;
float hum;
String tempC;

//--------------------Water Level Sensor pins---------------//
#define sensorPower 7 //
#define sensorPin A0

//--------------------Floater Switch Sensor pins--------------//
#define FLOAT_SENSOR  2
int floater;

//-----------------ESP8266 Serial WiFi Module---------------//
#define IP "184.106.153.149"// IP of cloud thingspeak.com
#define SSID "CYTA7998" // "SSID-WiFiname"
#define PASS "jcxFzjgEdeDyfN6a" // "WIFIpassword"
String msg = "GET /update?key=3CUDH3CEV02NRIPA";

//-------------- Value for storing water level ----------------//
int val = 0;
int level;

//------------------ Values for millis - delay ------------------//
unsigned long DeviceStarted = 0;
const long interval = 1800000; //30minutes
```

```
updateButton = 0 ;

//--------------used for loop function-----------------------//
int error;
```

Σε αυτό το σημείο, γίνονται οι αρχικοποιήσεις όλων των μεταβλητών.

3. Συνάρτηση connectWifi()

```
boolean connectWiFi(){
  Serial.println("AT+CWMODE=1");
  delay(2000);
  String cmd="AT+CWJAP=\"";
  cmd+=SSID;
  cmd+="\",\"";
  cmd+=PASS;
  cmd+="\"";
  Serial.println(cmd);
  delay(5000);
  if(Serial.find("OK")){
    return true;
  }else{
    return false;
  }
}
```

Με την συνάρτηση αυτή επιτυγχάνεται η σύνδεση της συσκευής με το internet.
Συγκεκριμένα χρησιμοποιούνται οι εντολές:
I. Η εντολή AT+CWMODE=mode (όπου mode: 1-3) ορίζει την λειτουργία του WI-FI.

- Αν mode = 1, τότε η λειτουργία είναι Station (σταθμός). Όταν η συσκευή βρίσκεται σε λειτουργία σταθμού, αυτό σημαίνει ότι είναι συνδεδεμένη σε ασύρματο router. Δηλαδή, σε αυτή την λειτουργία οι συσκευές συμπεριφέρονται σαν προγράμματα-clients.
- Αν mode = 2, τότε η λειτουργία είναι Soft Access Point (Soft AP). Αυτή η λειτουργία δημιουργεί ένα ασύρματο δίκτυο στη συσκευή, το οποίο στη συνέχεια μπορεί να χρησιμοποιηθεί από άλλες συσκευές ώστε να συνδεθούν και να έχουν για πρόσβαση στο Διαδίκτυο. Με πιο απλά λόγια, η συσκευή μετατρέπεται σε Host. (π.χ. WiFi HotSpot).
- Αν mode = 3, τότε η συσκευή λειτουργεί σαν Station και σαν Soft AP.

Το mode που χρησιμοποιήθηκε ήταν το mode =1, γιατί η συσκευή είναι client που συνδέεται σε wireless router.

ΙΙ. Η εντολή AT+CWJAP= “SSID”, “PASS” χρησιμοποιείται ώστε να συνδεθεί η συσκευή στο router/Wifi.. Η μεταβλητή SSID έχει αρχικοποιηθεί με το username του router ενώ το PASS με τον κωδικό του router.

4. Συνάρτηση readWaterSensor()

```
int readWaterSensor() {
  digitalWrite(sensorPower, HIGH);  // Turn the sensor ON
  delay(10);              // wait 10 milliseconds
  val = analogRead(sensorPin);    // Read value from sensor
  digitalWrite(sensorPower, LOW);   // Turn the sensor OFF
  return val;             // send value
}
```

Η συνάρτηση αυτή καλείται, ώστε να πάρει μέτρηση το Water Level Sensor. Όπως αναφέρθηκε στο κεφάλαιο 3.1.2, ο αισθητήρας αυτός ενεργοποιείται μόνο όταν καλείται αυτή η συνάρτηση και στην συνεχεια απενεργοποιείται. Έτσι προσδίδει στον αισθητήρα περισσότερο χρόνο ζωής.

5. Συνάρτηση updateValues()

```
void updateValues(){
  String cmd = "AT+CIPSTART=\"TCP\",\"";
  cmd += IP;
  cmd += "\",80";
  Serial.println(cmd);
  delay(2000);
  if(Serial.find("Error")){
    return;
  }
  cmd = msg ;
  cmd += "&field1=";     //field 1 for temperature
  cmd += String(temp);
  cmd += "&field2=";  //field 2 for humidity
  cmd += String(hum);
  cmd += "&field3=";  //field 3 for water level
  cmd += String(level);
  cmd += "&field4=";  //field 4 for floater
  cmd += String(floater);
```

```
  cmd += "\r\n";
  Serial.print("AT+CIPSEND=");
  Serial.println(cmd.length());
  if(Serial.find(">")){
    Serial.print(cmd);
  }
  else{
    Serial.println("AT+CIPCLOSE");
    //Resend
     error=1;
  }
}
```

Σε αυτήν την συνάρτηση χρησιμοποιούνται δύο AT command:

- AT+CIPSTART = “TCP”, “184.106.153.149”, “80”

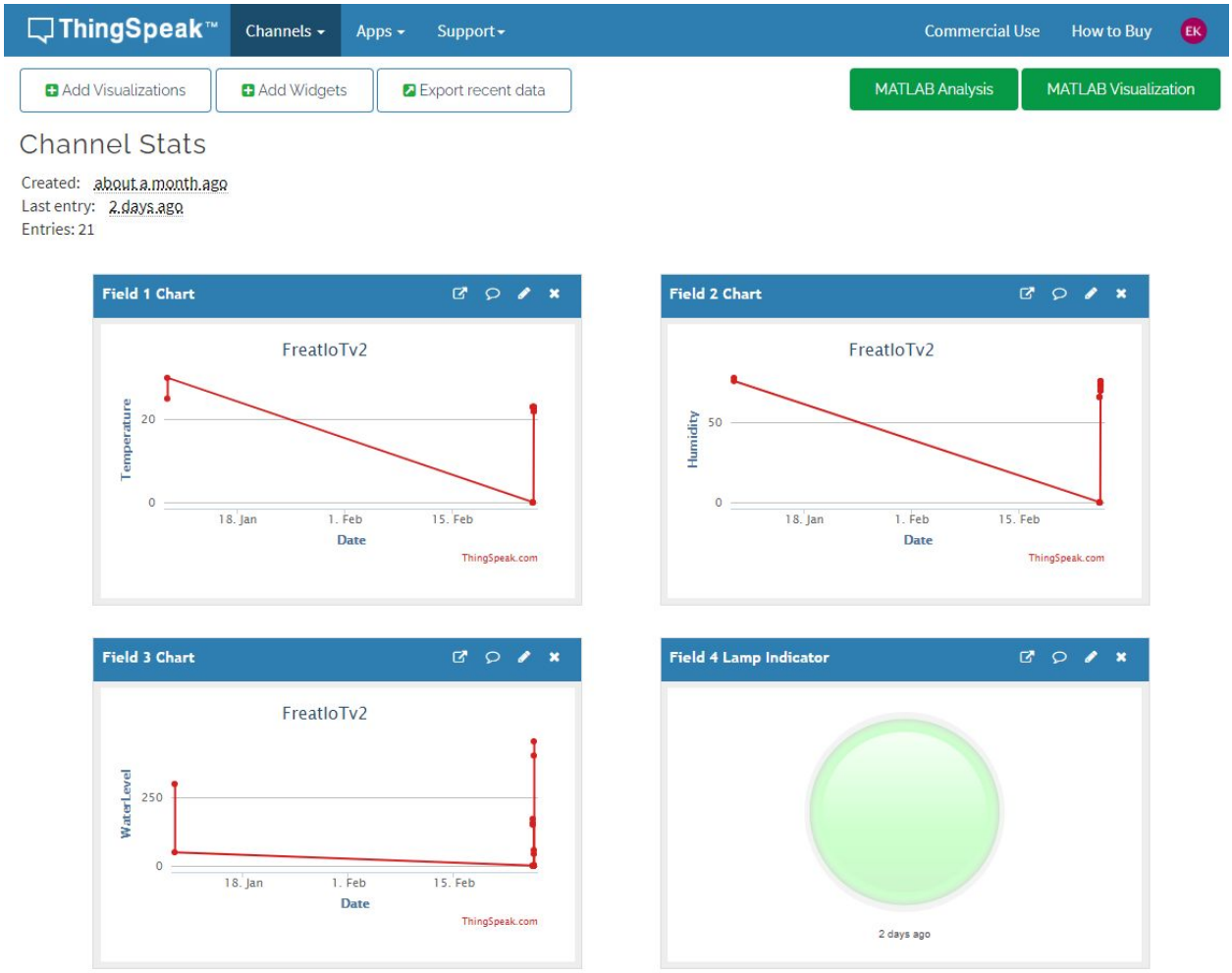
  Η παραπάνω εντολή συνδέει το ESP8266, ως TCP client, με το IP 184.106.153.149 (που είναι το IP του cloud service) και το αντίστοιχο port που είναι 80.
  Σε περίπτωση που βρεθεί κάποιο σφάλμα (Serial.find("Error")) όταν εκτελείται το AT command, τότε διακόπτεται ολόκληρη η συνάρτηση.

- AT+CIPSEND = GET /update?key=LRJG3B3EJMPEPGG5&field1={θερμοκρασία} &field2 ={υγρασία}&field3={στάθμη νερού}&field4={φλοτέρ}

  Η παραπάνω εντολή χρησιμοποιείται για να στείλουμε δεδομένα πάνω στο TCP connection. Ουσιαστικά η εντολή αυτή περιέχει ένα ‘GET’ http request, το οποίο χρησιμοποιεί το API “/update?key=LRJG3B3EJMPEPGG5&field1={θερμοκρασία} &field2 ={υγρασία}&field3={στάθμη νερού}&field4={φλοτέρ}” ώστε να στείλει τις μετρήσεις των αισθητηρίων στο cloud service. Το cloud service που χρησιμοποιήθηκε είναι της thingspeak [16]

Στέλνοντας το AT command AT+CIPSEND στο ESP, το response είναι “>”. Αν πάρουμε το response αυτό τότε τα δεδομένα αποστέλλονται επιτυχώς στο cloud. Αν, όμως δεν πάρει το σωστό response, τότε αποστέλλεται το AT command “AT+CIPCLOSE” ώστε να διακοπεί το TCP connection . Επίσης, το error flag παίρνει την τιμή 1, για να γίνει επανεκτέλεσης όλης της διαδικασίας.

Στο cloud, το dashboard έχει την παρακάτω μορφή:



Εικόνα 3.12: Dashboard του Cloud Service
(Ιστοσελίδα: https://thingspeak.com/channels/1279772)

3.2.2. Λειτουργία κώδικα

Στο κεφάλαιο 3.2.1 εξηγήθηκαν όλες οι συναρτήσεις, αλλά για να δούμε πως λειτουργεί end-to-end η συσκευή, δεν έχουμε παρά να κοιτάξουμε τις συναρτήσεις setup() και loop().

Συνάρτηση setup()

```
void setup() {
  Serial.begin(115200);
  //WaterLevel
  pinMode(sensorPower, OUTPUT); // Set D7 as an OUTPUT
  digitalWrite(sensorPower, LOW);
  //Floater
  // initialize the pushbutton pin as an input:
```

```
 pinMode(FLOAT_SENSOR, INPUT_PULLUP);

 do{
   Serial.println("AT");
    delay(5000);
 }while(!Serial.find("OK")) ;

 delay(5000);
 if(Serial.find("OK")){
  connectWiFi();
 }

}
```

Η συνάρτηση setup() πραγματοποιείται μόνο μία φορά όταν τεθεί η συσκευή σε λειτουργία. Παρακάτω θα αναλύσουμε τι κάνει η κάθε εντολή:

- Serial.begin(115200).
  Αυτή η εντολή χρησιμοποιείται ώστε να ενεργοποιηθεί το Serial Monitor του Arduino. Ενεργοποιώντας το, το Arduino μπορεί να στείλει εντολές (AT commands) στον ESP. Το 115200 είναι το baud rate/speed όπου στέλνονται οι χαρακτηρες.
- pinMode(sensorPower, OUTPUT);
  Με αυτή την εντολή, θέτουμε την ψηφιακή είσοδο D2 (του Water Level Sensor) ως έξοδος.
- pinMode(FLOAT_SENSOR, INPUT_PULLUP);
  Ενεργοποιούνται τα internal pullup resistors του φλοτέρ, που σημαίνει ότι ο αισθητήρας επίπλευσης ειναι στο HIGH.
- do{
    Serial.println("AT");
  }while(!Serial.find("OK"))

  Η εντολή “AT” είναι δοκιμαστική. Αφού στείλουμε αυτή την εντολή περιμένουμε το OK σαν response, για να βεβαιωθούμε ότι ο ESP δουλεύει. Όποτε στέλνουμε την εντολή AT μέχρι να λάβουμε την απάντηση.

- if(Serial.find("OK")){
    connectWiFi();
  }

  Εφόσον το AT εντολή επιστρέφει OK σαν response, καλείται η συνάρτηση connectWiFI() ώστε να συνδεθεί το ESP στο διαδίκτυο.

Συνάρτηση loop()

```
void loop() {
  unsigned long currentMillis = millis();
  if  (currentMillis - DeviceStarted >= interval)
  {
     DeviceStarted = currentMillis;

     start: //label
     error=0;

     //Read temperature and humidity values from DHT sensor:
     temp = DHT.temperature;
     hum = DHT.humidity;
     level = readWaterSensor();

     if(digitalRead(FLOAT_SENSOR) == LOW)
     {
       floater=0;
     }
     else
     {
       floater=1;
     }

     updateValues();

     //Resend if transmission is not completed
     if (error ==1){
       goto start; //go to label "start"
     }
  }
    if (floater == 1 || level> 500)
    {
       interval = 10000 //10 seconds
    }
    else
  {
   interval=60000; // 10 minutes
   }
}
```

Η συνάρτηση loop() είναι η συνάρτηση που τρέχει επανάληψη.

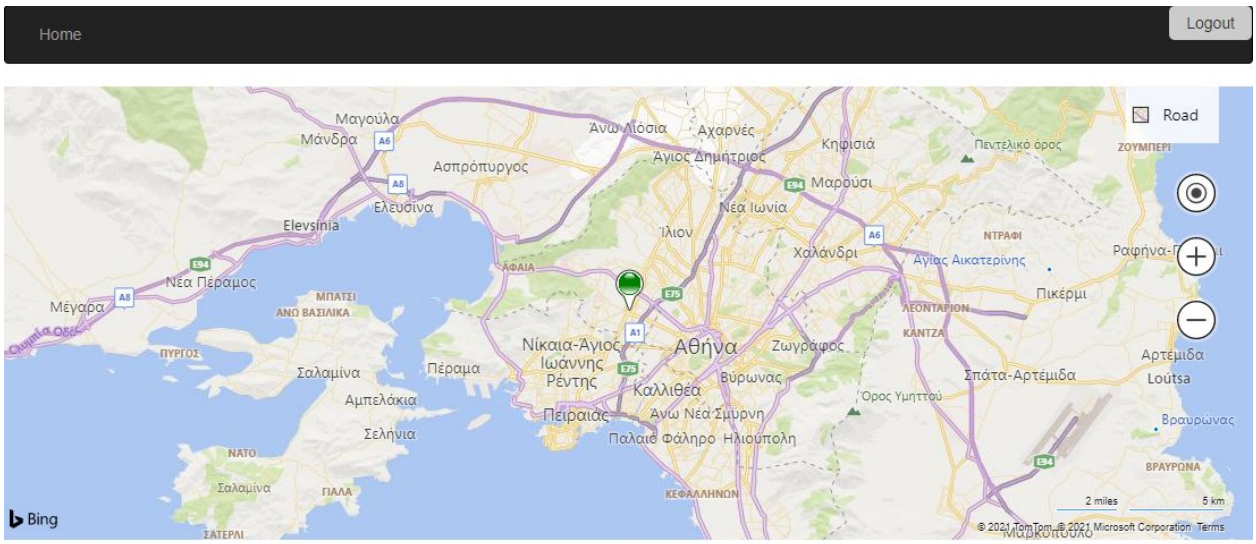
Η λογική που έχει χρησιμοποιηθεί είναι:

1. Θέτουμε την μεταβλητή currentMillis, η οποία παίρνει τιμή από την συνάρτηση millis(). Η συνάρτηση millis() επιστρέφει τον αριθμό των millisecond που έχουν περάσει από την στιγμή που τέθηκε η συσκευή σε λειτουργία.
2. Η συνθήκη if (currentMillis - DeviceStarted >= interval) σημαίνει ότι για να εκτελεστεί ότι βρίσκεται μέσα στην if, πρέπει η συνθήκη "currentMillis - DeviceStarted >= interval" να είναι TRUE. Οι μεταβλητές έχουν τις τιμές currentMillis = millis(), DeviceStarted=0ms και interval = 1800000 ms (30 λεπτά). Συνεπώς η συνθήκη αυτή γίνεται TRUE, κάθε 30 λεπτά.
3. Όταν γίνει η συνθήκη TRUE, η μεταβλητή DeviceStarted παίρνει την τιμή του currentMillis, η οποία στην 1η επανάληψη θα είναι 1800000 ms.
4. Το label "start" είναι το σημείο όπου ξεκινάνε οι αισθητήρες να παίρνουν μέτρηση. Αρχικά το DHT11 (μεταβλητές temp και hum), όπου μετράει την θερμοκρασία και υγρασία. Εν συνεχεία, το Water Level (μεταβλητή level), μετράει την στάθμη του νερού και τέλος το floater (μεταβλητή floater), μετράει αν έχει βουλώσει το φρεάτιο.
5. Αφού ολοκληρωθούν οι μετρήσεις, καλείται η συνάρτηση updateValues(), με την οποία αποστέλλονται οι μετρούμενες τιμές στο cloud.
6. Σε περίπτωση που υπάρξει κάποιο σφάλμα ένω αποστέλλονται οι μετρούμενες τιμές στο cloud, τότε μέσα στη συνάρτηση updateValues() η μεταβλητή error γίνεται 1. Συνεπώς, στην loop συνάρτηση, η συνθήκη if (error==1) γίνεται TRUE και εκτελείται η εντολή "goto start;". Εκτελώντας αυτή την εντολή,τρέχει ξανά ο κώδικας ξεκινώντας από το "start" label.
7. Κι τέλος, σε περίπτωση που το φλοτέρ έχει την τιμή 1 ή το water level έχει τιμή πάνω από 500, τότε θέλουμε η συσκευή να παίρνει πιο συχνές μετρήσεις. Συνεπώς το interval παίρνει την τιμή των 2 λεπτών.

3.3 Interface

Η εικόνα παρακάτω παρουσιάζει το interface της συσκευής. Αποτελείται από έναν χάρτη που περιέχει ένα εικονίδιο, το οποίο συμβολίζει την συσκευή. Επίσης κάτω από τον χάρτη,υπάρχει ένας πίνακας όπου αναφέρονται όλα τα στοιχεία της συσκευής και οι μετρήσεις.. Θα αναλυθεί το κάθε σκέλος του interface στην συνέχεια του κεφαλαίου. Μπορείτε να δείτε την ιστοσελίδα σε αυτήν την διεύθυνση https://freatiot.000webhostapp.com. (διαπιστευτήρια: Όνομα χρήστη = paspartou , κωδικός = paspartou)

Να σημειωθεί ότι στο Παράρτημα Α, Β και Γ θα αναλυθεί ένα μεγάλο κομμάτι του κώδικα κι επίσης στην ιστοσελίδα https://github.com/ellez1514/ptixiaki-v1 μπορεί να βρεθεί ολόκληρος ο κώδικας της σελίδας.



Πληροφορίες Φρεατίου	Σφάλματα	User Messages	Demo

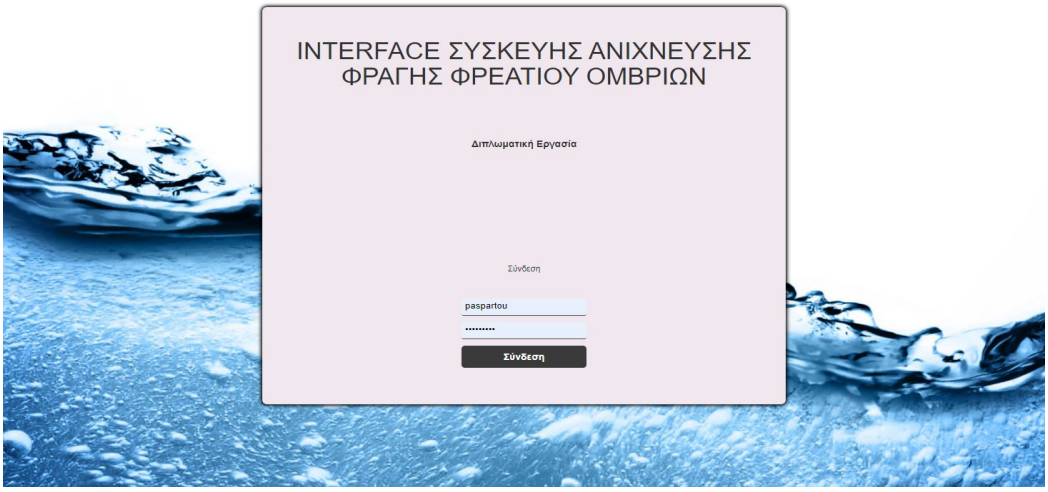
ID	81
ID Συσκευής	imei24
Υγρασία	4
Αισθητήρας Στάθμης Νερού	0
Αισθητήρας Floater	0
Τελευταία Ενημέρωση	2021-02-25 01:12:46

Εικόνα 3.13: Ιστοσελίδα

3.3.1. Σελίδα Login

Το πρώτο πράγμα που συναντά ο χρήστης που επισκέπτεται την ιστοσελίδα είναι η login σελίδα. Η ιστοσελίδα αυτή θα χρησιμοποιείται από εξουσιοδοτημένα άτομα, όπως δήμους ή ιδιώτες, συνεπώς η είσοδος πρέπει να επιτρέπεται μόνο με την χρήση διαπιστευτηρίων. Η λογική

είναι αρκετά απλή. Σε κάθε χρήστη θα δίνονται συγκεκριμένα credentials, τα οποία σώζονται στη βάση. Ο χρήστης, μπορεί να συνδεθεί μόνο αν χρησιμοποιήσει έγκυρα credentials, αλλιώς θα πάρει μήνυμα λάθους. Με την γλώσσα HTML και CSS φτιάχτηκε το interface που φαίνεται στην εικόνα 3.14. Με την χρήση της γλώσσας PHP, επιτυγχάνεται η πιστοποίηση του χρήστη. Πιο αναλυτικά, όταν ο χρήστης εισάγει τα διαπιστευτήρια, τρέχει μια εντολή sql για να εντοπιστούν στην βάση. Αν εντοπιστούν επιτυχώς, τότε ο χρήστης εισέρχεται στην πλατφόρμα, ενώ στην αντίθετη περίπτωση εμφανίζεται ένα μήνυμα λάθους, που απεικονίζεται στην εικόνα 3.15.



Εικόνα 3.14: Login Page

Αξίζει να σημειωθεί ότι η ιστοσελίδα λειτουργεί με PHP Session. Το PHP Session δημιουργεί μοναδικό αναγνωριστικό(id) σε καθε browser για αναγνώριση του χρήστη. [17]. Τι σημαίνει αυτό;
Σημαίνει ότι ένας χρήστης δεν μπορεί να εισέλθει στην σελίδα με κανέναν τρόπο αν δεν έχει το id, που αυτό αποκτάται μόνο με την σύνδεση των σωστών διαπιστευτηρίων.

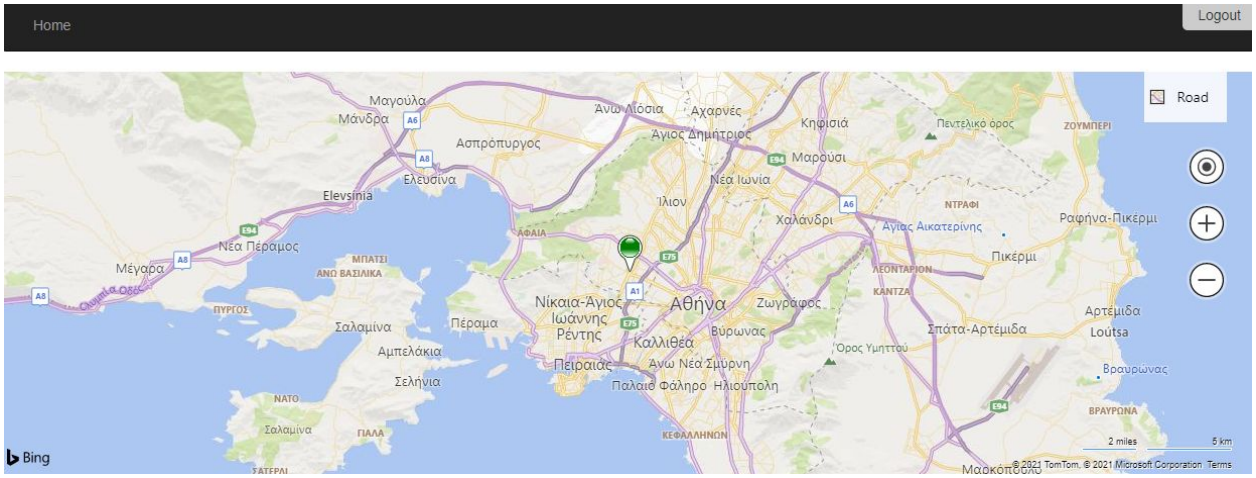
Να σημειωθεί ότι τα σωστά διαπιστευτήρια για την είσοδο στην σελίδα είναι:
Όνομα χρήστη: paspartou
Κωδικός: paspartou

Εικόνα 3.15: Login Page με χρήση λάθων διαπιστευτηρίων

3.3.2. Χάρτης

Με την χρήση των σωστών διαπιστευτηρίων, ο χρήστης εισέρχεται εντός της σελίδας όπου απεικονίζεται ένας χάρτης και ένας πίνακας. Ο χάρτης που χρησιμοποιήθηκε είναι το Bing Maps [18].

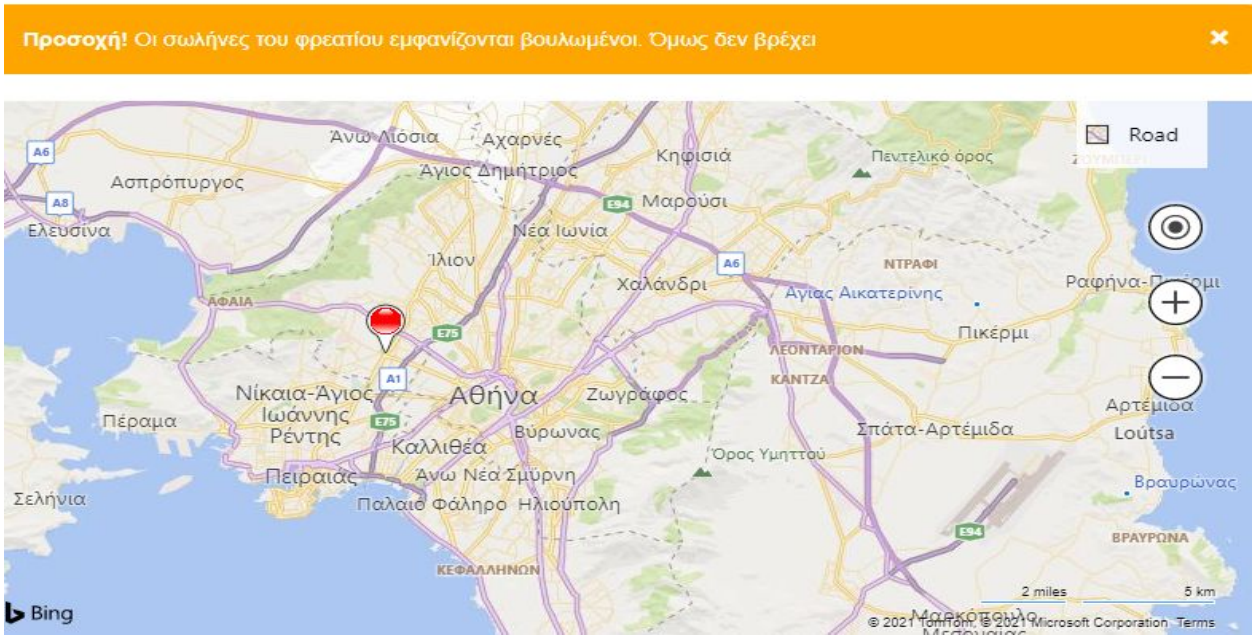
Όπως φαίνεται στο χάρτη, έχει τοποθετηθεί ένα σημάδι με πράσινο κύκλο, που αντιπροσωπεύει την συσκευή. Συγκεκριμένα, ο πράσινος κύκλος, σημαίνει ότι η συσκευή είναι σε καλή κατάσταση, δηλαδή ότι δεν υπάρχει κάποιο σφάλμα στο φρεάτιο.



Εικόνα 3.16: Χάρτης με συσκευή χωρίς σφάλμα

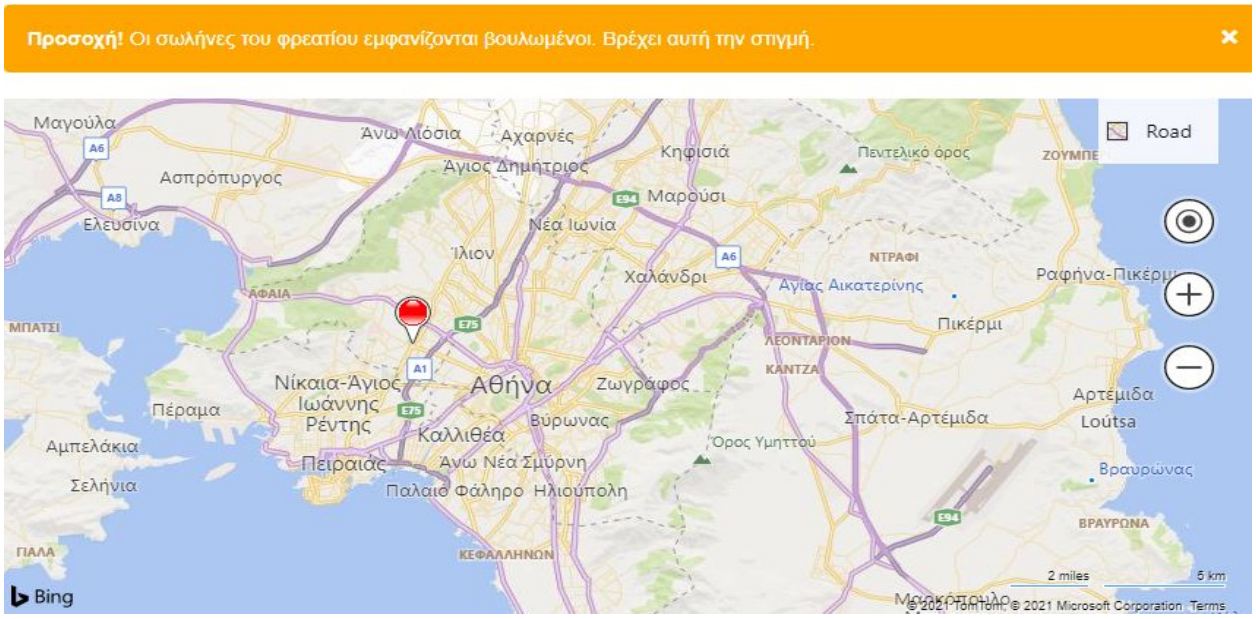
Η συμπεριφορά της σελίδας αλλάζει σε περίπτωση σφάλματος. Το σημάδι που αντιπροσωπεύει την συσκευή αλλάζει χρώμα και τα μηνύματα που λαμβάνονται κατά την διάρκεια ενός σφάλματος, ποικίλουν αναλόγως με τις καιρικές συνθήκες ή το αισθητήριο το οποίο παρουσιάζει το σφάλμα. Παρακάτω θα δούμε την συμπεριφορά του αισθητήρα επίπλευσης/φλοτέρ και τα μηνύματα που εμφανίζονται ανάλογα με το αν βρέχει ή όχι.

Για τον αισθητήρα επίπλευσης, αν παρουσιαστεί κάποιο σφάλμα και η τιμή του γίνει 1 τότε το το σημάδι θα αλλάξει χρώμα από πράσινο σε κόκκινο. Σε περίπτωση που δεν βρέχει τότε απεικονίζεται το μήνυμα της εικόνας 3.17. Ουσιαστικά, από πλευράς κώδικα, ελέγχουμε ένα web API , το οποίο παρέχεται από την ιστοσελίδα Open Weather Maps [19] και μας επιστρέφει τις καιρικές συνθήκες της περιοχής που έχουμε ορίσει. Αν το API μας επιστρέψει ότι ο καιρός είναι ηλιόλουστος, τότε εμφανίζεται το παρακάτω μήνυμα.
Σε αυτήν την περίπτωση σημαίνει ότι ο αισθητήρας βούλωσε λόγω ενός εξωτερικού παράγοντα.



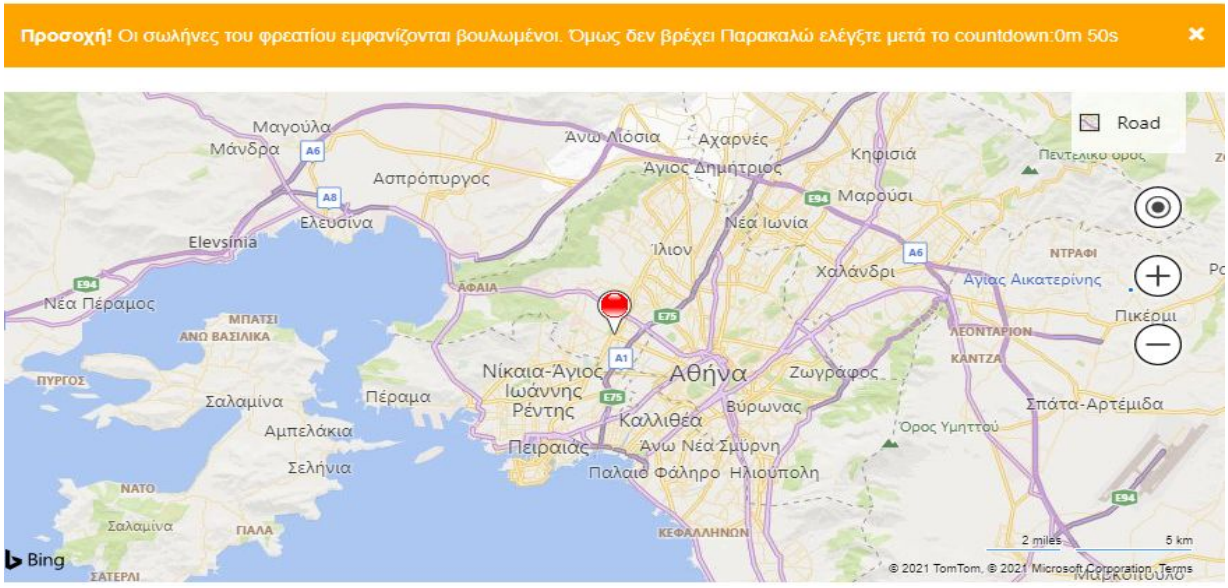
Εικόνα 3.17: Χάρτης με σφάλμα φλοτέρ όταν δεν βρέχει

Αν το API του καιρού μας επιστρέψει ότι αυτή την στιγμή βρέχει, τότε το μήνυμα που εμφανίζεται στην σελίδα είναι το παρακάτω.



Εικόνα 3.18: Χάρτης με σφάλμα φλοτέρ ενώ βρέχει

Όπως αναφέρθηκε στο κεφάλαιο 2.1, δεν μπορούμε να διαπιστώσουμε αν ένα φρεάτιο είναι φραγμένο κατά την διάρκεια μιας βροχής. Θα πρέπει να περιμένουμε 30 λεπτά αφού τελειώσει η βροχή, ώστε να αποφανθεί αν είναι φραγμένο ή όχι. Συνεπώς σε αυτή την περίπτωση, καλούμε το API του καιρού μέχρι να επιστρέψει ότι σταμάτησε να βρέχει.
Όταν το API του καιρού επιστρέψει ότι δεν βρέχει πλέον, τότε εμφανίζεται το παρακάτω μήνυμα:

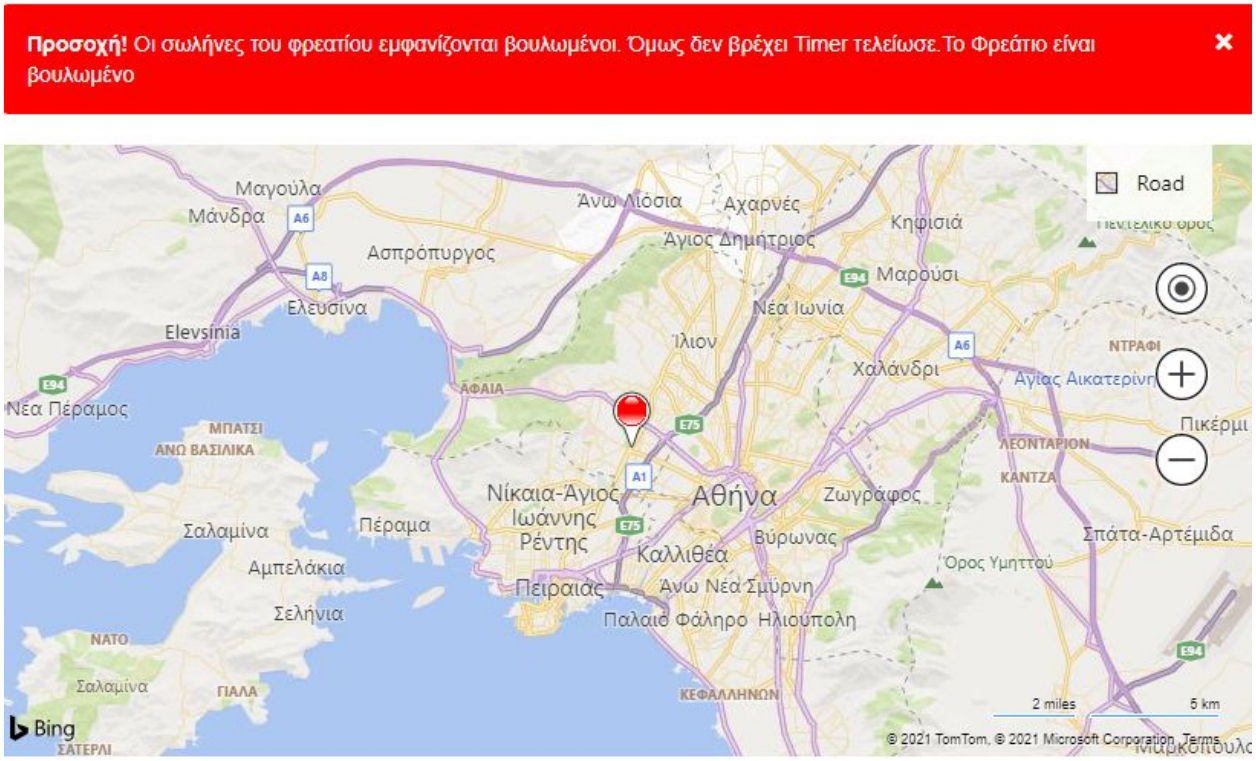


Εικόνα 3.19: Χάρτης με σφάλμα φλοτέρ όταν τελειώσει να βρέχει

Όπως φαίνεται στη παραπάνω εικόνα, εμφανίζεται στο μήνυμα ένα timer. Αυτό το timer, είναι ο χρόνος των 30 λεπτών που απαιτείται για να κρίνουμε αν έχει βουλώσει το φρεάτιο ή όχι. Η λογική που ακολουθείται είναι η εξής:

1. Αποθηκεύεται η ώρα που σταμάτησε η βροχή κι προστίθενται 30 λεπτά σε αυτήν.
2. Η ώρα με τα προσαυξημένα 30 λεπτά, σώζεται στην βάση δεδομένων.
3. Ξεκινάει το countdown, όπου συγκρίνεται η τωρινή ώρα με την προσαυξημένη ώρα που έχει αποθηκευτεί στην βάση.

Όταν τελειώσουν τα 30 λεπτά, αν το αισθητήριο επίπλευσης συνεχίσει να έχει την τιμή 1 τότε σημαίνει ότι το φρεάτιο είναι φραγμένο. Σε αυτήν την περίπτωση, το παρακάτω μήνυμα εμφανίζεται. Επίσης, στην βάση αποθηκεύεται το σφάλμα.



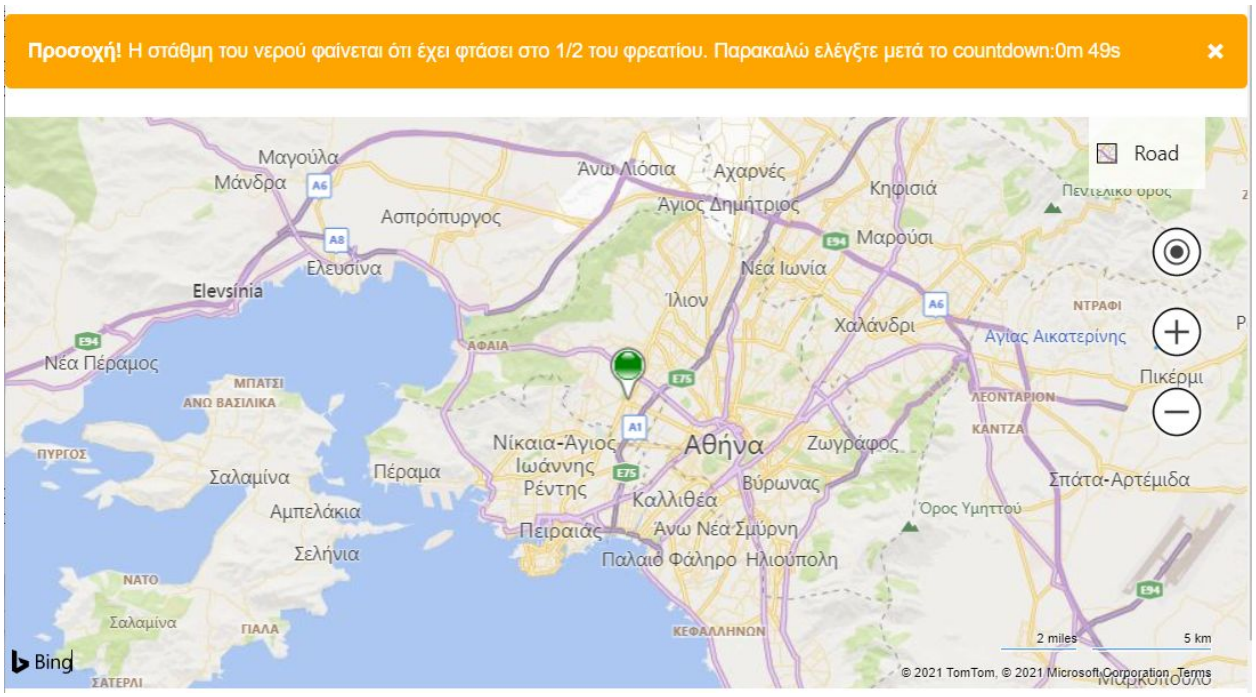
Εικόνα 3.20: Χάρτης με σφάλμα φλοτέρ μετά το timer

Για τον αισθητήρα στάθμης νερού δεν υπάρχει η διαφοροποίηση των μηνυμάτων ανάλογα με τις καιρικές συνθήκες.

Σε περίπτωση που η στάθμη φτάσει στη μέγιστη τιμή του (που είναι 533), δηλαδή όταν η στάθμη του νερού φτάσει στη μέση του φρεάτιο, τότε εμφανίζεται το μήνυμα λάθους της εικόνας 3.21. Στην περίπτωση αυτή, ο κύκλος παραμένει πράσινος (ο κύκλος κοκκινίζει μόνο όταν το floater έχει πάρει την τιμή 1, που σημαίνει ότι το φρεάτιο έχει φραγεί). Επιπλέον, εμφανίζεται ένα timer. Για πιο λόγο, όμως, μπήκε timer εδώ για τον αισθητήρα στάθμης νερού;

Ας υποθέσουμε ότι κάποιος καθαρίζει το πεζοδρόμιο κι πετάξει νερό μέσα στο φρεάτιο. Τότε το

αισθητήριο νερού θα ενεργοποιηθεί (διότι ενώ πέφτει το νερό, η στάθμη θα εμφανιστεί ως 500 έστω και για λίγα δευτερόλεπτα). Την στιγμή που θα ενεργοποιηθεί το αισθητήριο, τότε στην βάση αυτό θα γραφτεί ως σφάλμα, πράγμα που δεν ισχύει. Για να μην αποθηκεύονται, λοιπόν, false positives στην βαση, χρησιμοποιήθηκε το timer. Άρα, μόλις ενεργοποιηθεί το αισθητήριο (η στάθμη είναι πάνω από 500), τότε ενεργοποιείται ένα countdown 20 λεπτών. Αν μετά την πάροδο των 20 λεπτών, η στάθμη συνεχίσει να είναι πάνω από 500, τότε θεωρούμε ότι όντως υπάρχει νερό στο φρεάτιο.



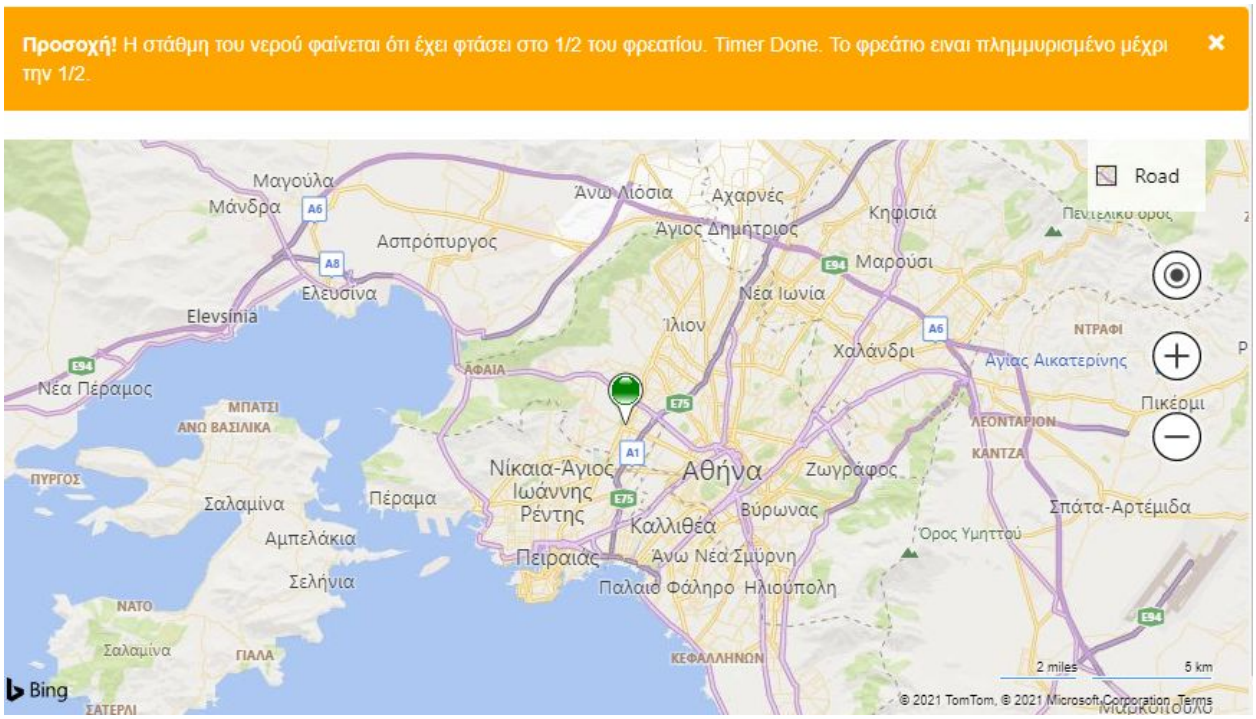
Εικόνα 3.21: Χάρτης με σφάλμα στάθμης νερού

Το timer που χρησιμοποιείται αισθητήρα στάθμης έχει την ίδια λογική με το timer του αισθητήρα επίπλευσης. Όταν ενεργοποιηθεί ο αισθητήρας, τότε στην τωρινή ώρα προστίθενται 20 λεπτά και αποθηκεύεται στην βάση. Στην συνέχεια, ξεκινά το countdown συγκρίνοντας την τωρινή ώρα με την ώρα που είναι αποθηκευμένη στην βάση.

Αξίζει να τονιστεί ότι αν ο αισθητήρας στάθμης νερού εμφανίζεται με την τιμή 500 μετά την πάροδο των 20 λεπτών, αλλά ο αισθητήρας επίπλευσης έχει την τιμή 0, τότε σημαίνει ότι είτε:

- Ο αισθητήρας επίπλευσης δεν δουλεύει σωστά.
- Έχει κολλήσει κάτι πάνω στον αισθητήρα στάθμης νερού και για αυτό έχουμε false positive.

Αφού τελειώσει το countdown, τότε εμφανίζεται το μήνυμα της παρακάτω εικόνας



Εικόνα 3.22: Χάρτης με σφάλμα στάθμης νερού μετά το timer

3.3.3. Πληροφορίες συσκευής

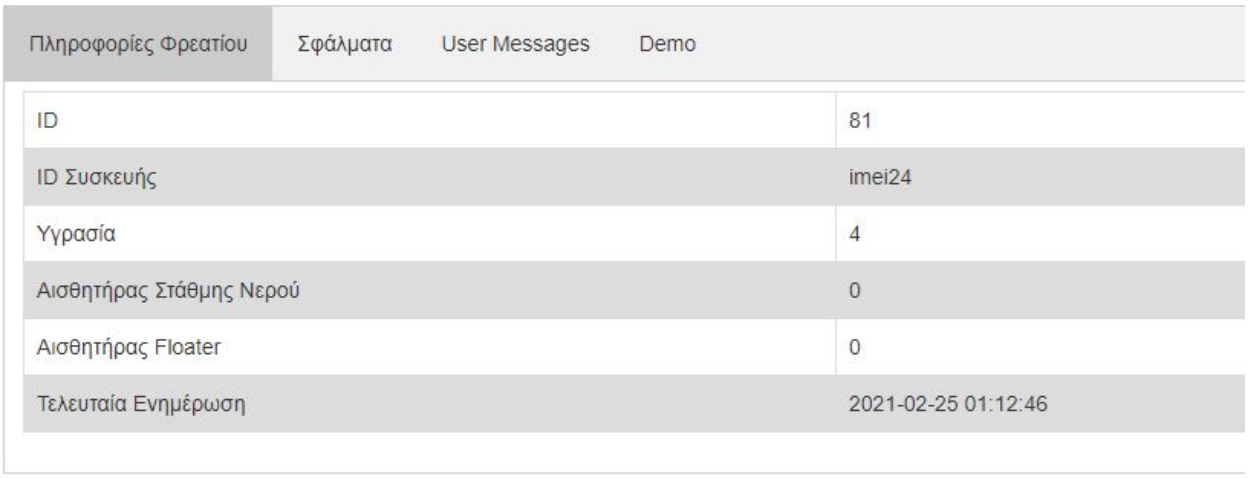
Όπως είδαμε παραπάνω, στο υποκεφάλαιο 3.2.2, η συσκευή παίρνει μέτρηση κάθε 30 λεπτά. Η μέτρηση αυτή αποθηκεύεται, αρχικά, στο cloud και στην συνέχεια, στην βάση δεδομένων της ιστοσελίδας. Αυτές οι μετρήσεις, εμφανίζονται στον πίνακα "Πληροφορίες Φρεατίου".

Από πλευράς κώδικα, υπάρχει μια συνθήκη που συγκρίνει κάθε 10 λεπτά το "last updated date" που είναι αποθηκευμένο στην βάση με την τελευταία ημερομηνία της μέτρησης που είναι αποθηκευμένη στο Cloud. Όταν η ημερομηνία που υπάρχει στο cloud είναι νεότερη από την υπάρχουσα ημερομηνία της βάσης δεδομένων, τότε οι νέες μετρήσεις καταγράφονται στην βάση δεδομένων και στην συνέχεια εμφανίζονται στον πίνακα "Πληροφορίες Φρεατίου". Να σημειωθεί, ότι λόγω της τεχνολογίας του AJAX, δεν χρειάζεται να γίνει refresh για να εμφανιστούν οι καινούργιες μετρήσεις στον πίνακα. Γίνεται αυτόματα.

Οι πληροφορίες που εμφανίζονται στον πίνακα είναι:

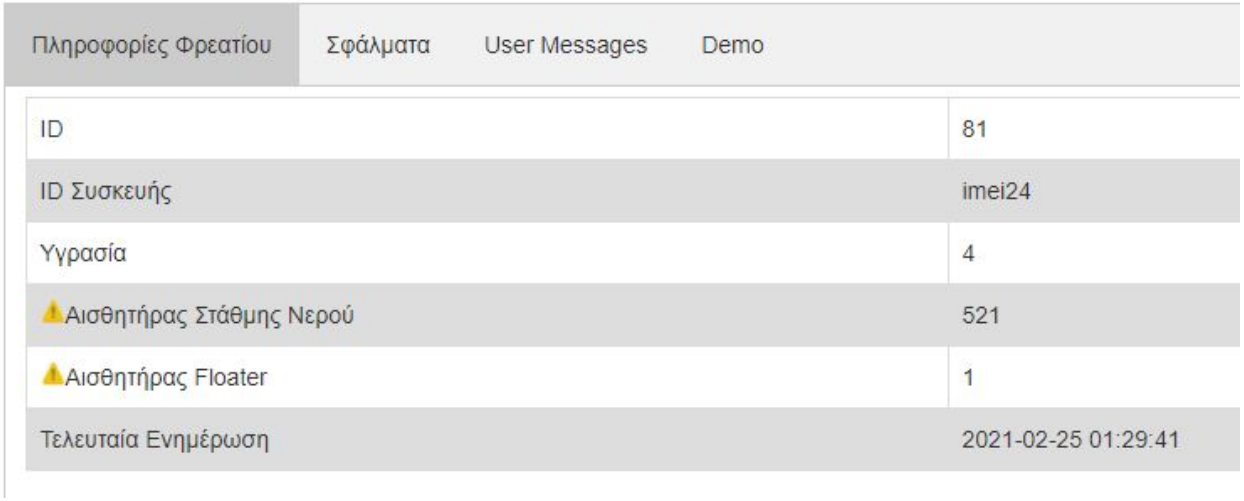
- Id
- ID Συσκευής, όπου κάθε συσκευή έχει δικό του id
- Υγρασία
- Αισθητήρας Στάθμης Νερού
- Αισθητήρας Επίπλευσης / Floater

- Τελευταία Ενημέρωση, όπου είναι η ημερομηνία που πήραμε την τελευταία μέτρηση των αισθητηρίων



Εικόνα 3.23: Πληροφορίες συσκευής

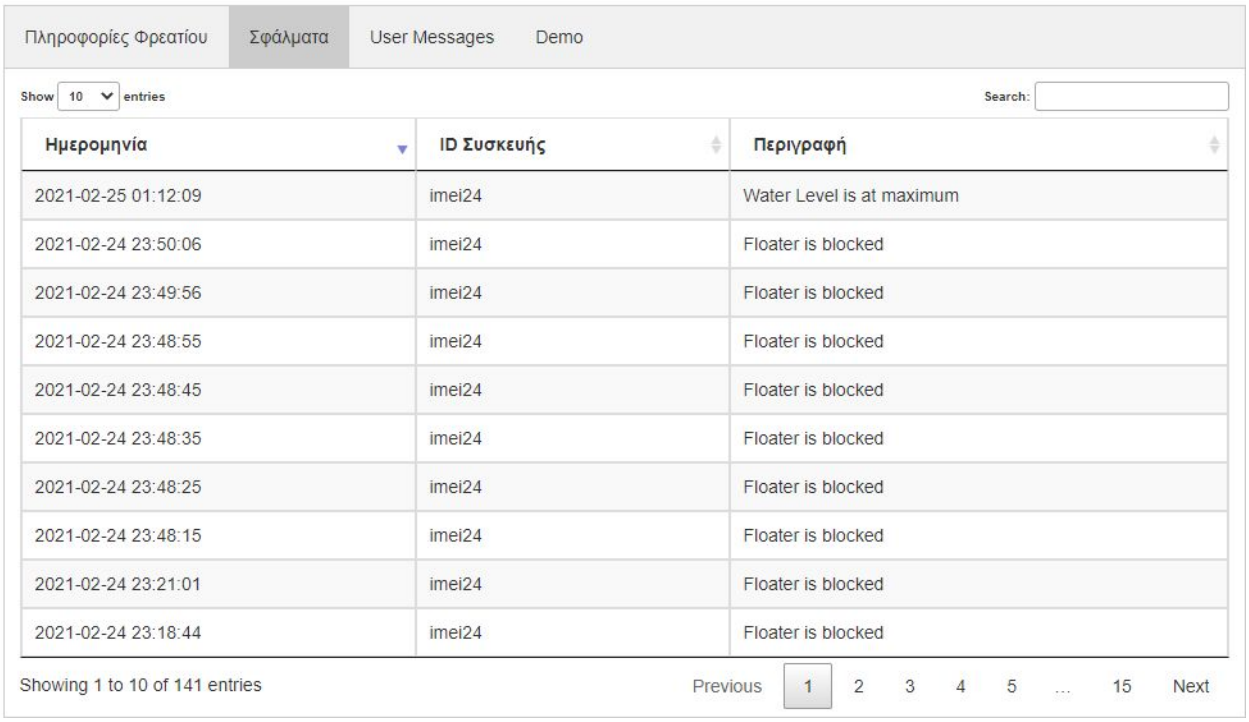
Σε περίπτωση σφάλματος στον αισθητήρα στάθμης νερού ή στον αισθητήρα επίπλευσης / Floater τότε στον πίνακα εμφανίζεται το εικονίδιο του “Warning”.



Εικόνα 3.24: Πληροφορίες συσκευής με σφάλμα

Στον 2ο πίνακα εμφανίζονται όλα τα σφάλματα που έχουν υπάρξει στην συσκευή . Οι πληροφορίες που εμφανίζονται είναι:

- Ημερομηνία
- ID Συσκευής
- Περιγραφή σφάλματος



Εικόνα 3.25: Σφάλματα συσκευής

Υπάρχουν σφάλματα δύο ειδών:

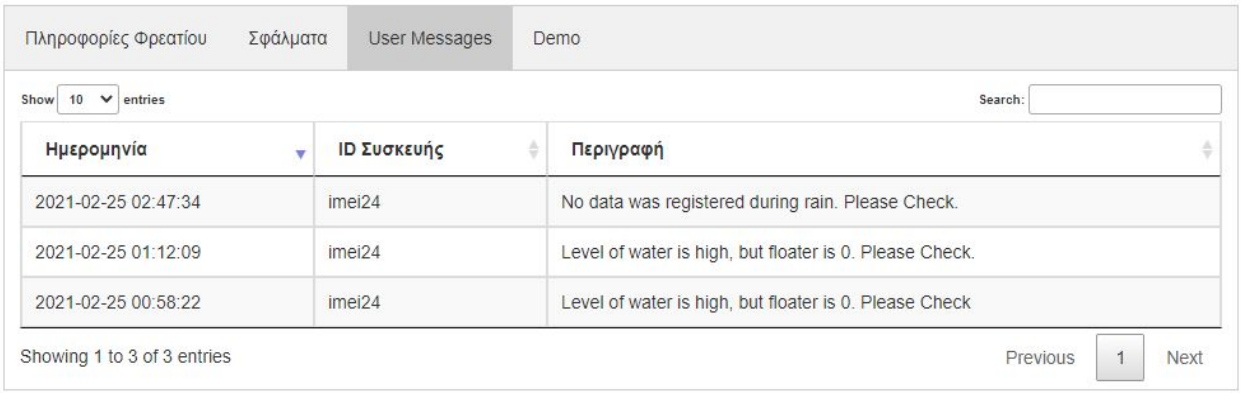
1. Floater is blocked, που σημαίνει ότι μετά το τέλος της βροχής και την πάροδο των 30 λεπτών, ο αισθητήρας επίπλευσης εξακολουθεί να έχει την τιμή 1, άρα ο σωλήνας του φρεατίου έχει βουλώσει.
2. Water Level is at maximum, που σημαίνει ότι μετά την πάροδο των 20 λεπτών, ο αισθητήρας στάθμης εξακολουθεί να έχει τιμή μεγαλύτερη από 500, άρα το φρεάτιο είναι πλημμυρισμένο.

Οι τιμές του πίνακα αυτού, ανανεώνονται κάθε 30 λεπτά με την χρήση της τεχνικής Ajax.

0 3ος πίνακας ονομάζεται User Messages. Εκεί βλέπουμε μηνύματα που αφορούν την συσκευή, τα οποία όμως δεν μπορούν να χαρακτηριστούν ως σφάλματα, αλλά ως δυσλειτουργίες. Για παράδειγμα:

1. Σε περίπτωση που ο αισθητήρας στάθμης νερού έχει τιμή μεγαλύτερη από 500 (η στάθμη του νερού φτάνει περίπου το μισό του φρεατίου), θα πρέπει ο αισθητήρας επίπλευσης να έχει την τιμή 1 διότι το φρεάτιο θα είναι πλημμυρισμένο. Αν ο αισθητήρας δεν έχει την τιμή 1, τότε σημαίνει ότι πιθανόν κάτι δεν δουλεύει σωστά με τον αισθητήρα επίπλευσης και πρέπει να ελεγχθεί, συνεπώς εμφανίζεται το user msg “Level of water is high, but floater is 0. Please Check.”

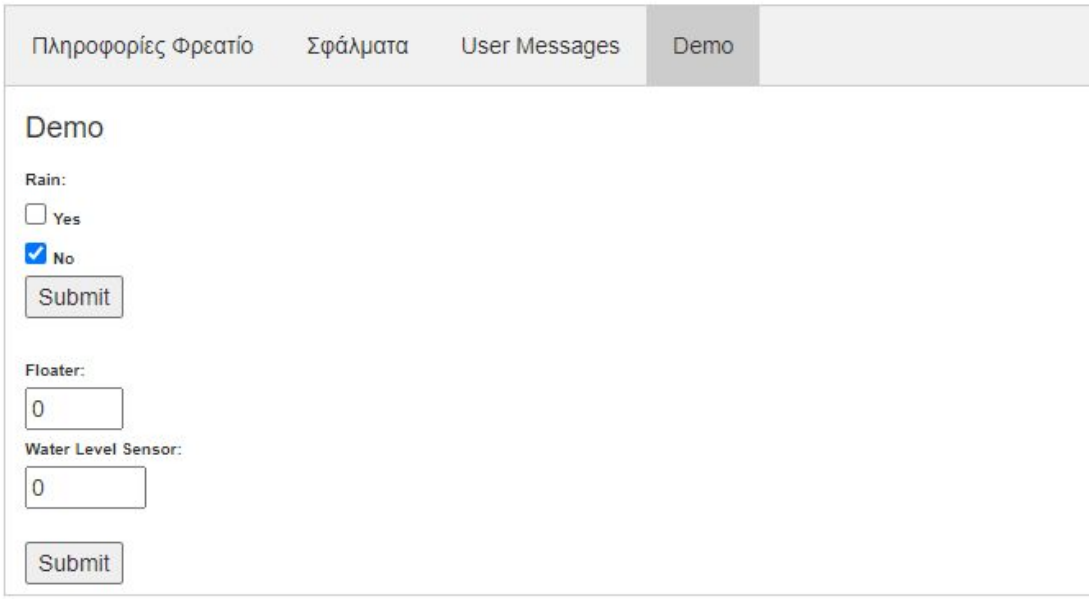
2. Έστω ότι βρέχει για 20 λεπτά. Το νερό θα κυλάει πάνω στον αισθητήρα στάθμης νερού που σημαίνει ότι θα καταγραφούν μετρήσεις. Όμως μετά το πέρας της βροχής, αν στην βάση δεν έχει καταγραφεί καμία μέτρηση κατά την διάρκεια της βροχής, αυτό σημαίνει δύο πράγματα, είτε το αισθητήριο δεν δουλεύει σωστά ή οι σχάρες του φρεατίου είναι μπλοκαρισμένες και το νερό δεν εισχωρεί εντός του φρεατίου, συνεπώς εμφανίζεται το μήνυμα "No data was registered during rain. Please Check."
   Από πλευράς κώδικα για να επιτευχθεί αυτό, ελέγχουμε το API που μας επιστρέφει το status του καιρού. Μόλις τελειώσει η βροχή, το API θα επιστρέψει το status Clear, οπότε τσεκάρουμε στην βάση το άθροισμα των τελευταίων 30-50 δεδομένων του αισθητήρα στάθμης νερού και επίπλευσης. Αν το άθροισμα και των δύο είναι μηδέν τότε σημαίνει ότι το νερό δεν εισέρχεται στο φρεάτιο.



Εικόνα 3.26: User Messages

Ο 4ος και τελευταίος πίνακας ονομάζεται Demo. Προστέθηκε για τα πλαίσια της διπλωματικής αυτής, έτσι ώστε να μπορεί να γίνει μία εύκολη παρουσίαση της λογικής που έχει η ιστοσελίδα. Όπως φαίνεται από την παρακάτω εικόνα, δίνονται οι επιλογές:

- Rain (τιμές: YES, NO), όπου αν η τιμή είναι YES σημαίνει ότι αυτή τη στιγμή βρέχει ενώ αν η τιμή είναι NO τότε δεν βρεχει,
- Floater (τιμές: 0 και 1), όπου αν η τιμή είναι 0 τότε ο σωλήνας του φρεατίου δεν είναι βουλωμένος ενώ αν είναι 1 τότε είναι.
- Water Level Sensor (τιμές: 0 εώς 533) , όπου αν τιμή είναι κάτω από 500 τότε το water level δεν ενεργοποιείται, ενώ όταν είναι μεγαλύτερο ή ίσο με 500 τότε το φρεάτιο θεωρείται πλημμυρισμένο.



Εικόνα 3.27: Demo

Το σενάριο που πρέπει να ακολουθηθεί προκειμένου να αναπαραχθούν οι συμπεριφορές που αναφέρθηκαν παραπάνω είναι:

Σενάριο: Συσκευή χωρίς σφάλμα

1. Αφήνουμε όλες τις default τιμές, όπως φαίνονται στην παραπάνω εικόνα ((Rain = No, Floater = 0 και Water Level Sensor = 0) και πατάμε submit.
    - Το αποτέλεσμα θα είναι αυτό που απεικονίζεται στην εικόνα 3.16, δηλαδή ο κύκλος στον χάρτη θα έχει πράσινο χρώμα και δεν θα υπάρχει κανένα σφάλμα.

Σενάριο: Αισθητήρας Επίπλευσης

1.Στο floater βάζουμε την τιμή 1, αφήνουμε τις άλλες default τιμές (Rain = No και Water Level Sensor = 0) και πατάμε submit.
    - Το αποτέλεσμα θα είναι αυτό που απεικονίζεται στην εικόνα 3.17, δηλαδή ο κύκλος στον χάρτη θα κοκκινίσει και θα εμφανιστεί μήνυμα που θα αναφέρει ότι δεν βρέχει, αλλά το φρεάτιο είναι βουλωμένο.

2. Στο floater αφήνουμε την τιμή 1, αλλάζουμε το Rain σε Yes και πατάμε submit.
    - Το αποτέλεσμα θα είναι αυτό που απεικονίζεται στην εικόνα 3.18, δηλαδή ο κύκλος στον χάρτη παραμένει κόκκινος και θα εμφανιστεί μήνυμα που θα αναφέρει ότι το φρεάτιο εμφανίζεται βουλωμένο.

3.Αλλάζουμε την τιμή του Rain σε No και αφήνουμε την τιμή του floater στο 1. Αυτό σημαίνει ότι σταμάτησε να βρέχει αλλά ο αισθητήρας είναι ακόμα βουλωμένος.
    - Το αποτέλεσμα θα είναι αυτό που απεικονίζεται στην εικόνα 3.19, δηλαδή ο κύκλος στον χάρτη παραμένει κόκκινος και θα εμφανιστεί μήνυμα που θα περιέχει το timer. Για

λόγους παρουσίασης το timer έχει την τιμή του ενός λεπτού. Μόλις τελειώσει το timer, το μήνυμα σφάλματος θα κοκκινίσει και θα αναφέρει ότι το timer τελείωσε και το φρεάτιο είναι βουλωμένο, όπως φαίνεται στην εικόνα 3.20. Επίσης, στον πίνακα "Σφάλματα" θα καταγραφεί και θα απεικονιστεί το σφάλμα "Floater is blocked".

4. Αλλάζουμε την τιμή του floater στο 0. Τότε η συσκευή παύει να επιστρέφει σφάλμα και ο κύκλος στον χάρτη πρασινίζει.

Σε κάθε αλλαγή που γίνεται στην τιμή του αισθητήρα επίπλευσης, ο πίνακας "Πληροφορίες Φρεατίου" θα ανανεώνεται.

Σενάριο: Αισθητήρας Στάθμης νερού

Όπως αναφέρθηκε παραπάνω, ο αισθητήρας στάθμης νερού δεν εξαρτάται από τις καιρικές συνθήκες, έτσι λοιπόν στο Demo πίνακα, θα αλλάξουμε μόνο την τιμή του Water Level Sensor.

1.Στο Water Level Sensor βάζουμε τιμή πάνω από ή ίσο με 500 (αυτή η τιμή έχει οριστεί για να θεωρείται πλημμυρισμένο το φρεάτιο). Αφήνουμε τις άλλες default τιμές (Rain = No και Floater = 0) και πατάμε submit.

- Το αποτέλεσμα θα είναι αυτό που απεικονίζεται στην εικόνα 3.21, δηλαδή ο κύκλος στον χάρτη παραμένει πράσινος και θα εμφανιστεί μήνυμα αναφέρει ότι το φρεάτιο είναι πλημμυρισμένο και θα περιέχει το timer. Για λόγους παρουσίασης το timer έχει την τιμή του ενός λεπτού. Μόλις τελειώσει το timer, το μήνυμα σφάλματος θα κοκκινίσει και θα αναφέρει ότι το timer τελείωσε και το φρεάτιο είναι πλημμυρισμένο, όπως φαίνεται στην εικόνα 3.22. Επίσης, στον πίνακα "Σφάλματα" θα καταγραφεί και θα απεικονιστεί το σφάλμα "Water Level is at Maximum".

2.Αλλάζουμε την τιμή του Water Level Sensor σε τιμή κάτω από 500. Τότε η συσκευή παύει να επιστρέφει σφάλμα.

Σε κάθε αλλαγή που γίνεται στην τιμή του αισθητήρα στάθμης νερού, ο πίνακας "Πληροφορίες Φρεατίου" θα ανανεώνεται.

Σενάριο: User Message "Level of water is high, but floater is 0. Please Check"

Όπως αναφέρθηκε και παραπάνω, αν η στάθμη του νερού φτάσει στην μέση του φρεατίου και δεν ενεργοποιηθεί ο αισθητήρας επίπλευσης, σημαίνει ότι υπάρχει μια δυσλειτουργία. Για να δούμε αυτή την συμπεριφορά, στο Demo πίνακα, θα αλλάξουμε μόνο την τιμή του Water Level Sensor σε 501 και θα περιμένουμε να τελειώσει το timer. Μόλις τελειώσει το timer, θα γραφτεί στον πίνακα του User Messages "Level of water is high, but floater is 0. Please Check"

ΚΕΦΑΛΑΙΟ 4: ΜΕΛΕΤΗ ΠΡΟΒΛΕΠΤΙΚΟΥ ΑΛΓΟΡΙΘΜΟΥ

Αν, υποθετικά, 50 συσκευές λειτουργούν σε ένα διάστημα 2-3 μηνών, τότε θα συλλέγονταν εκατοντάδες μετρήσεις. Αυτά τα δεδομένα, θα μπορούσαν να αξιοποιηθούν ώστε να εξαχθούν σημαντικά συμπεράσματα. Βγάζοντας τα σωστά συμπεράσματα από αυτά τα δεδομένα, μπορούν να γίνουν απαραίτητες αλλαγές ώστε να διορθωθούν διάφορα θέματα όπως στον σχεδιασμό των φρεατίων. Για αυτόν τον λόγο, αποφασίστηκε να γίνει θεωρητική μελέτη ενός προβλεπτικού αλγορίθμου. Συγκεκριμένα, το σενάριο που θα μελετηθεί είναι "Ανάλογα με διάφορους παράγοντες όπως γεωμορφολογία, χιλιοστά βροχής, αριθμό φρεατίων που έχουν φραγεί κτλ, μπορεί να προβλεφθεί μια πλημμύρα;"

Το σενάριο αυτό είναι αρκετά περίπλοκο, διότι υπάρχουν πάρα πολλοί παράγοντες που μπορούν να οδηγήσουν σε μία πλημμύρα, όπως:

- Η περιοχή ή γεωμορφολογία. Κάθε περιοχή έχει διαφορετική γεωμορφολογική μορφή, δηλαδή κάποιες περιοχές βρίσκονται κοντά σε ποτάμια, κάποιες κοντά σε βουνό και κάποιες άλλες σε κατηφορικό σημείο. Όπως είναι φυσικό, περιοχές που έχουν αυτά τα στοιχεία, προφανώς, θα είναι πιο επιρρεπείς στις πλημμύρες.
- Η εποχή. Η εποχή είναι ένας πολύ σημαντικός παράγοντας. Για παράδειγμα το φθινόπωρο πέφτουν φύλλα. Αν υποθέσουμε ότι βρέξει και τα φύλλα δεν έχουν καθαριστεί, τότε πολύ πιθανό να κολλήσουν στο σωλήνα του φρεατίου. Επίσης τον φθινόπωρο και χειμώνα βρέχει περισσότερο, που κι αυτός είναι ένας σημαντικός παράγοντας.
- Πότε καθαρίστηκε ο δρόμος τελευταία φορά. Ας πάρουμε για παράδειγμα την εποχή. Κάθε φθινόπωρο πέφτουν τα φύλλα των δέντρων. Αν αυτά δεν καθαριστούν εγκαίρως υπάρχει κίνδυνος φραγής των φρεατίων. Άλλο ένα παράδειγμα είναι με τα δημόσια ή ιδιωτικά έργα. Σε περίπτωση που δεν καθαριστούν τα μπάζα κι πέσει βροχή, μπορεί να προκαλέσει φραγή στις σχάρες και στον σωλήνα υδροσυλλογής του φρεατίου.
- Φρεάτια ομβρίων υδάτων. Προφανώς αν όλα τα φρεάτια της περιοχής είναι φραγμένα, τότε προφανώς το νερό της βροχής δεν έχει πουθενά αλλού να πάει, συνεπώς οι πιθανότητες πλημμύρας αυξάνονται δραματικά.
- Ο βασικότερος παράγοντας, ίσως, είναι τα χιλιοστά βροχής και η ώρα της βροχής. Προφανώς αν πέσουν 5mm νερού (σε 1 ώρα) υπάρχει πολύ μικρότερη πιθανότητα πλημμύρας σε σχέση με το να πέσουν 60mm(σε 1 ώρα).

Σίγουρα, υπάρχουν κι άλλοι παράγοντες πέρα από τους πέντε που αναφέρθηκαν πιο πάνω, αλλά αυτά τα δεδομένα αρκούν ώστε να βρεθούν οι πιθανότητες πλημμύρας.

Στον παρακάτω πίνακα φαίνεται κάποια στοιχεία που συλλέχθηκαν από την ΕΜΥ [20] και από την διαδικτυακή εφημερίδα Εθνικός Κήρυξ[21], τα οποία δείχνουν διάφορα στοιχεία που προκάλεσαν πλημμύρα.

Location	mm of rain	hours of rain	mm of rain in 1h	Month	Season	Geomorfological	Flood
Agios Panteliomonas	72	10	40	November	Fall	River Overflow	1
Athens	80	4		November	Fall	No	1
Neo Faliro	75	8		November	Fall	No	1
Mosxato	60	7		November	Fall	No	1
Peristeri	114	8	40	November	Fall	River Overflow	1
Peristeri	80	9		November	Fall	River Overflow	1
Tzitzifies	66	4		November	Fall	River Overflow	1
Tauros	78	8		november	Fall	River Overflow	1
Mosxato	70	5		november	Fall	River Overflow	1
Malakassa	50	4		February	Winter	Mountain	1
Mandra	60	24		November	Fall	River Overflow	1

Πίνακας 4.1: Δεδομένα με πλημμύρες (Πηγή: ΕΜΥ και Εθνικός Κήρυξ)

Αρχικά εύκολα μπορούμε να παρατηρήσουμε ότι η εποχή που πραγματοποιήθηκαν οι περισσότερες πλημμύρες ήταν το φθινόπωρο (τον μήνα Νοέμβριο) πράγμα που σημαίνει ότι η εποχή παίζει πολύ σημαντικό ρόλο. Αμά κοιτάξει κανείς τα κλιματικά δεδομένα της Ελλάδας ανά μήνα [22], θα δει ότι τα περισσότερα χιλιοστά βροχής πέφτουν, συνήθως, το φθινόπωρο και τον χειμώνα (Νοέμβριο και Δεκέμβριο) κάτι το οποίο φαίνεται κι από τον πίνακα παραπάνω, οπότε βγάζει λογική να έχουμε περισσότερες πλημμύρες εκείνη την εποχή.

.

Κάτι άλλο που μπορούμε να βγάλουμε σαν συμπέρασμα από τον πίνακα είναι ότι στις περισσότερες περιοχές που πραγματοποιήθηκαν οι πλημμύρες κύριος παράγοντας ήταν η γεωμορφολογία τους. Δηλαδή μπορούμε να δούμε ότι στις περισσότερες περιπτώσεις η πλημμύρα ξεκίνησε λόγω της υπερχείλισης ποταμού.

Ένας ακόμη σημαντικός παράγοντας είναι πόση ώρα θα διαρκέσει η βροχή. Για παράδειγμα αν σε μία ώρα βρέχει πάνω από 70 χιλιοστά βροχής υπάρχει μεγάλη πιθανότητα πλημμύρας, αλλά αν ήταν 70 χιλιοστά στην διάρκεια τεσσάρων ωρών τότε ίσως οι σωλήνες του φρεατίου μπορούσαν να ανταπεξέλθουν στην εισερχόμενη ποσότητα του νερού.
Συνεπως, κρίνοντας αυτά τα δεδομένα του πίνακα, μπορεί να υπολογιστεί η πιθανότητα πλημμύρας.

4.1 Dataset

Το πρώτο και πιο σημαντικό βήμα για την δημιουργια ενός προβλεπτικού αλγορίθμου είναι να δημιουργηθεί μια κατάλληλη βάση δεδομένων. Η βάση δεδομένων ενός προβλεπτικού αλγόριθμου ονομάζεται dataset, το οποίο είναι ένα αρχείο excel, txt ή csv. Όπως τον πίνακα 4.1, έτσι κι το dataset πρέπει να περιέχει μια πληθώρα δεδομένων για την τοποθεσία, χιλιοστά και ώρες βροχή, γεωμορφολογία και την κατάσταση των φρεατίων. Πρέπει τα μισά δεδομένα που υπάρχουν στο dataset να έχουν οδηγήσει σε πλημμύρα (flood = 1) και τα άλλα μισά να μην έχουν (flood = 0), διότι το πρόβλημα που καλούμαστε να μελετήσουμε είναι να προβλέψουμε κατά πόσο θα συμβεί πλημμύρα ή όχι. Συνεπώς το dataset πρέπει να έχει true(έγινε πλημμύρα)

και false(δεν έγινε πλημμύρα) δεδομένα. Πρακτικά, για συλλέξουμε αξιόπιστα δεδομένα σχετικά με την κατάσταση των φρεατίων θα πρέπει να στήσουμε ένα συγκεκριμένο αριθμό συσκευών ανίχνευσης φραγής σε μια συγκεκριμένη περιοχή "υψηλού κινδύνου"(δηλαδή περιοχές στις οποίες έχουν πραγματοποιηθεί πλημμύρες στο παρελθόν) και να συλλεχθούν δεδομένα στην διάρκεια ενός εξαμήνου τουλάχιστον.

Τα δεδομένα σχετικά με τα χιλιοστά και τις ώρες βροχής μπορούν να συλλεχθούν από διάφορες σχετικές υπηρεσίες οι οποίες προσφέρουν open data, όπως η Εθνικό Μετεωρολογική Υπηρεσία (ΕΜΥ)[23] και στο Εθνικό Αστεροσκοπείο Αθηνών[24]. Επίσης, για κάθε περιοχή μπορούν να συλλεχθούν γεωμορφολογικά δεδομένα, δηλαδή αν η περιοχή βρίσκεται κοντά σε κατηφόρα/ανηφόρα, κοντά σε ποτάμι, κοντα σε βουνό κτλ. Τέλος, από τους δήμους μπορούν να συλλεχθούν το πότε καθαρίζονται οι δρόμοι κι αν πραγματοποιούνται έργα.

4.2. Αλγόριθμος πρόβλεψης

Όπως αναφέρθηκε παραπάνω, τα βήματα για την υλοποίηση του αλγόριθμου πρόβλεψης είναι ο ορισμός του προβλήματος και η συλλογή δεδομένων για το dataset. Το πρόβλημα που ορίστηκε είναι η πρόβλεψη πλημμύρας έχοντας ως δεδομένα την περιοχή, τα χιλιοστά και τις ώρες βροχής, το πότε καθαρίστηκε τελευταία φορά ο δρόμος, την εποχή και την κατάσταση των φρεατίων (δηλαδή πόσα φρεάτια ήταν βουλωμένα λίγο πριν ξεκινήσει η πλημμύρα). Το επόμενο βήμα είναι να υλοποιηθεί ο αλγόριθμος πρόβλεψης.

Για την επιλογή του σωστού αλγόριθμου, πρέπει να δούμε ποια θα είναι η έξοδος του συστήματος. Συμφωνά με το πρόβλημα που ορίστηκε, η έξοδος χωρίζεται σε δύο κατηγορίες, "θα πλημμυρίσει" ή "δεν θα πλημμυρίσει", άρα εύκολα μπορούμε να συμπεράνουμε ότι θα πρέπει να χρησιμοποιηθεί αλγόριθμος ταξινόμησης . Τι είναι, όμως, αλγόριθμος ταξινόμησης;

"Οι αλγόριθμοι ταξινόμησης στη μηχανική μάθηση χρησιμοποιούν δεδομένα εκπαίδευσης εισόδου για να προβλέψουν την πιθανότητα τα επόμενα δεδομένα να εμπίπτουν σε μία από τις προκαθορισμένες κατηγορίες. Μία από τις πιο κοινές χρήσεις της ταξινόμησης είναι το φιλτράρισμα των μηνυμάτων ηλεκτρονικού ταχυδρομείου σε "ανεπιθύμητα" ή "μη ανεπιθύμητα"."[25]

Υπάρχουν διάφοροι τύποι αλγορίθμων ταξινόμησης. Στην παρούσα διπλωματική θα δούμε τους παρακάτω 3 τύπους.

- Logistic Regression
- Decision Tree
- Support Vector Machine

Για την επιλογή του κατάλληλου αλγορίθμου, θα πρέπει να δοκιμάσουμε και τους 3

αλγορίθμους και να δούμε πιο μας δίνει την καλύτερη ακρίβεια. Η διαδικασία για να βρούμε την ακρίβεια γίνεται ως εξής:

1. Το dataset χωρίζεται σε δύο μέρη, το ένα κομμάτι αποτελεί το 80% του dataset και το άλλο κομμάτι το 20%
2. Ακολουθείται η διαδικασία της εκπαίδευσης στο 80% του dataset, όπου ουσιαστικά ο αλγόριθμος βρίσκει μοτίβα στα δεδομένα που αντιστοιχούν στην απάντηση που θέλουμε να προβλέψουμε και εξάγει ένα μοντέλο που αναγνωρίζει αυτά τα μοτίβα.
3. Αφού αναγνωριστούν τα μοτίβα αυτά, το μοντελο χρησιμοποιεί το υπόλοιπο 20% του dataset και προβλέπει την απάντηση του καθενός. Στην συνέχεια συγκρίνονται οι απαντήσεις που προέκυψαν από το μοντέλο με τις απαντήσεις που υπάρχουν στο dataset. Οι απαντήσεις του μοντέλου πρέπει να έχουν όσο το δυνατόν μικρότερο σφάλμα, συνεπάγοντας σε μεγαλύτερη ακρίβεια.
4. Στην συνέχεια το μοντέλο, θα μπορεί να δεχτεί νέες τιμές (που δεν βρίσκονται στο training dataset) και να τις κατηγοριοποιεί.

Τα προγραμματα που χρησιμοποιούνται για Machine Learning είναι Python, R, Matlab [26] και Excel.

Παρακάτω θα περιγραφούν οι 3 αλγόριθμοι ταξινόμησης, που αναφέρθηκαν παραπάνω.

4.2.1 Logistic Regression

Το logistic regression είναι ένας αλγόριθμος που χρησιμοποιείται κυρίως σε προβλήματα ταξινόμησης. Είναι ένας σχετικά απλός αλγόριθμος και δεν έχει πολύ δύσκολα μαθηματικά.

Χρησιμοποιεί γραμμική εξίσωση με ανεξάρτητες μεταβλητές (που πρόκειται για τα δεδομένα του dataset) για την πρόβλεψη μιας τιμής, όπως φαίνεται παρακάτω

$$z = \theta_0 + \theta_1 * x_1 + \theta_2 * x_2 + \dots \theta_n * x_n \qquad (4.2.1.1)$$

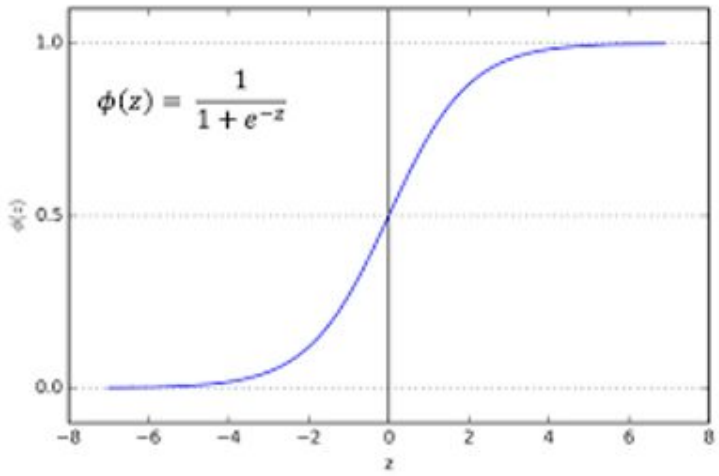
όπου n = ο αριθμός των δεδομένων

Αυτή η εξίσωση μπορεί να γραφτεί και με την μορφή εσωτερικού γινομένου:

$$z = \theta^T * x \qquad (4.2.1.2)$$

Όπου $\theta^T = [\ \theta_0 \quad \theta_1 \quad \theta_2 \dots\dots\dots \theta_n\ ]^T$ και $x = [1 \quad x_1 \quad x_2 \dots\dots x_n\ ]$

Το αποτέλεσμα της εξίσωσης, αυτής, πρέπει να είναι ανάμεσα στο 0 και στο 1, δηλαδή μέσα στο κλειστό σύστημα [0,1], διότι θέλουμε το αποτέλεσμα να ανήκει σε δύο κλάσεις είτε 0 είτε 1. Για να επιτευχθεί αυτό χρησιμοποιείται η σιγμοειδής συνάρτηση.



Εικόνα 4.1: Σιγμοειδής συνάρτηση (Πηγή - Ιστοσελίδα: https://www.quora.com/How-does-a-sigmoid-function-map-any-data-points-into-the-range-0-1)

Συνεπώς η σχέση 4.2.1.1 γίνεται ώς εξής

$$h = \varphi(z) = 1 / [1 + e^{-(\theta 0 + \theta 1 * x1 + \theta 2 * x2 + \ldots + \theta n * xn)}] \quad (4.2.1.3)$$

Η σιγμοειδής h ουσιαστικά επιστρέφει την πιθανότητα του αποτελέσματος να είναι 1. Όπως αναφέρθηκε παραπάνω, ο αλγόριθμος αυτός είναι ταξινόμησης οπότε το τελικό αποτέλεσμα πρέπει να είναι είτε 1-yes είτε 0-no. Συνεπως αν η σιγμοειδής επιστρέψει 0.7 τότε η πιθανότητα να είναι 1-yes είναι 0.7 (ή η πιθανότητα να είναι 0 είναι 0.3) ενώ αν επιστρέψει 0.3 η πιθανότητα να είναι 1-yes είναι 0.3(ή η πιθανότητα να είναι 0 είναι 0.7). Μπορούμε αλλιώς να πούμε αν h< 0.5 τότε ανήκει στη κλάση 0(δεν θα γίνει πλημμύρα) ενώ h>= 0.5 τότε θα θεωρείται ότι ανήκει στην κλάση 1(θα γίνει πλημμύρα).

Για να υπολογιστεί η ακρίβεια που προσφέρει ο logistic αλγόριθμος, χρησιμοποιείται το cost function, για να εκπαιδευτεί το μοντέλο με το 80% του dataset. Αυτή η διαδικασία ονομάζεται training/εκπαίδευση. Η συνάρτηση του cost μπορεί να προσδιοριστεί με τους δύο παρακάτω τρόπους:

$$Cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & \text{if } y = 1 \\ -log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases} \quad (4.2.1.4)$$

Εικόνα 4.2: Συνάρτηση κόστους (Πηγή - Ιστοσελίδα: Coursera Andrew Ng - ML Course https://www.coursera.org/learn/machine-learning)

Όπου $h_\theta(x)$ είναι το αποτέλεσμα της πρόβλεψης και y είναι το πραγματικό αποτέλεσμα.

Ή αν ενωθούν οι δύο παραπάνω συναρτήσεις(y=1 και y=0) , μπορεί να γραφτεί ως

$$J(\theta) = -\frac{1}{m}\sum\left[y^{(i)}\log(h\theta(x(i))) + \left(1-y^{(i)}\right)\log(1-h\theta(x(i)))\right] \quad (4.2.1.5)$$

Εικόνα 4.3:Συνάρτηση κόστους απλοποιημένη μορφή
(Πηγή - Ιστοσελίδα: Coursera Andrew Ng - ML Course https://www.coursera.org/learn/machine-learning)

Ουσιαστικά θέλουμε το cost function, να έχει όσο το δυνατόν μικρότερη τιμή, που συνεπάγεται σε μικρότερο σφάλμα, το οποίο οδηγεί σε μεγαλύτερη ακρίβεια του αλγόριθμου. Ο σκοπός της της εξίσωσης κόστους είναι να βρεθεί ένα τοπικό μέγιστο της καμπύλης J(θ). Συνεπώς , όπως, φαίνεται στην εικόνα 4.4, η εξίσωση gradient descent χρησιμοποιείται για αυτό τον σκοπό. Πραγματοποιούνται επαναλήψεις μέχρι να βρεθούν οι τιμές του θ που οδηγούν το J(θ) σε τοπικό μέγιστο. Σε κάθε επανάληψη το βήμα δεν πρέπει να είναι μεγάλο αλλιώς το τοπικό μέγιστο μπορεί να μην βρεθεί. Το learning rate a που απεικονίζεται στην εξίσωση της εικόνας 4.4, είναι το βήμα και η τιμή του επιλέγεται εμπειρικά (αν επιλεχθεί μια μικρη τιμή τότε θα πρέπει να γίνουν πολλές επαναλήψεις για να έχουμε σύγκλιση ενω αν η τιμή είναι πολύ μεγάλη τοτε το J(θ) μπορεί να μην μειώνεται σε κάθε επανάληψη και να μην συγκλίνει ποτέ).

Want $\min_\theta J(\theta)$:

Repeat {

$$\theta_j := \theta_j - \alpha\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

(simultaneously update all θ_j)

} (4.2.1.6)

Εικόνα 4.4:Gradient Descent Simplified
(Πηγή - Ιστοσελίδα: Coursera Andrew Ng - ML Course https://www.coursera.org/learn/machine-learning)

Στο πρόγραμμα Matlab αυτό θα γραφόταν ως εξής:

```
dataset = load('freatiotMat.csv');
x = dataset(:,1:end-1);
y = dataset(:,end);
m = length(x);

% fill x array with 1 and create theta array
x = [ones(length(x),1), x];
theta = zeros(size(x,2),1);

% Set the learning rate to 0.1
alpha = 0.1;
%Set epochs to 10000
for i = 1:100000
    % logistic function
    z = x * theta;
    % sigmoid function
    h = 1 ./ (1 + exp(-z));

    %error diff between model and dataset values
    %We want this to be 0
    error = h - y;

    % Calculate the gradient
    for j = 1:length(theta)
        gradient = 1/m * (h - y)' * x(:,j);
        % Update theta parameters
        theta(j) =  theta(j) - alpha * gradient;
    end


    % Calculate cost
    cost(i) = 1/m * ( -y' * log(h) - (1-y)' * log(1-h) );
end

% Displaying the final theta's
theta
```

Εικόνα 4.5: Logistic Regression Matlab (πρόγραμμα Matlab Online)

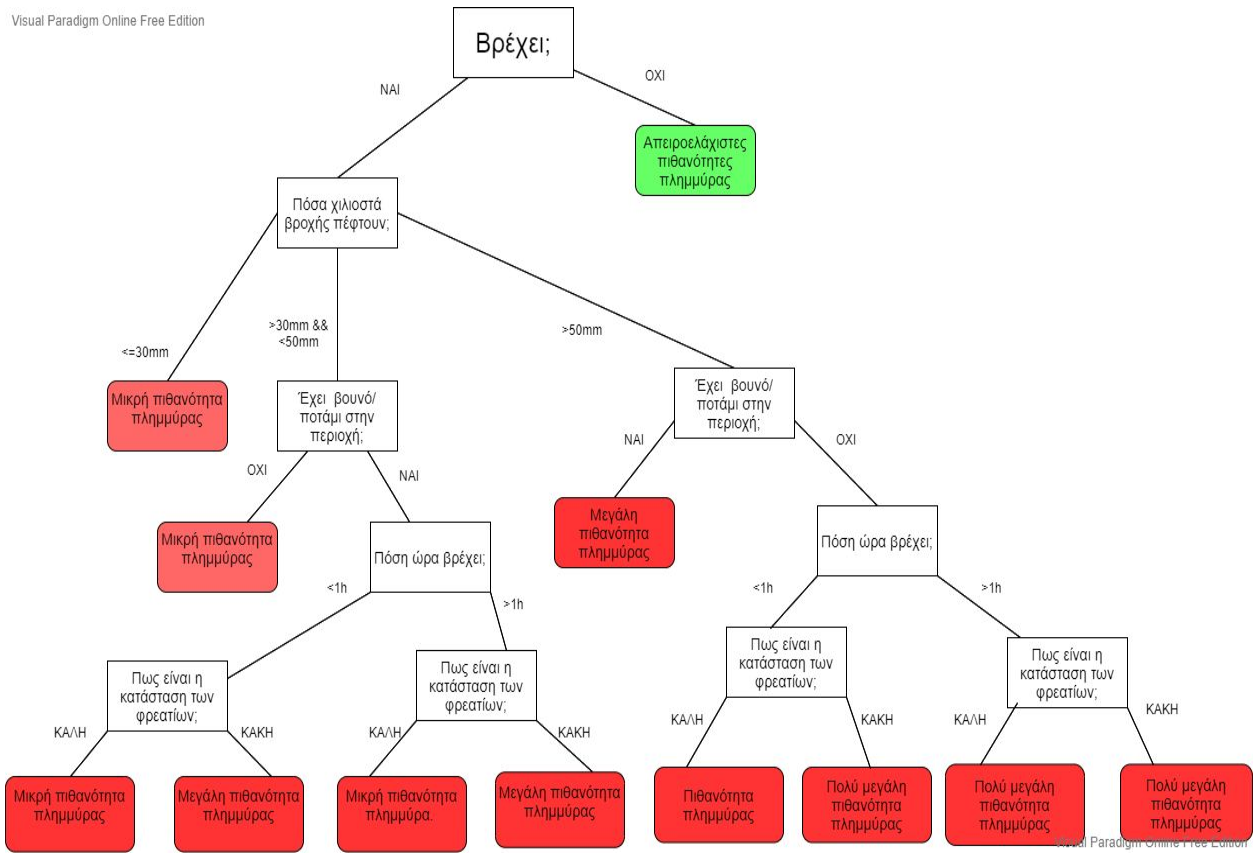
1.Πρώτα θα γίνει upload του dataset στο workspace του Matlab και στην συνέχεια καλείται στο script.
2. Ορίζονται οι είσοδοι x , που είναι όλες οι στήλες του dataset εκτός από την τελευταία.
3. Ορίζονται η έξοδος y, που είναι η τελευταία στήλη του dataset.
4. Για να δημιουργηθεί η εξίσωση του logistic, χρησιμοποιείται ο τύπος του εσωτερικού γινομένου (4.2.1.2) και για αυτό το το x και το theta παίρνουν την κατάλληλη μορφή.
5.Στην συνέχεια επιλέγεται εμπειρικά μια τιμή για το a και τον αριθμό των επαναλήψεων.
6. Ακολουθούν οι σχέσεις (4.2.1.2) και (4.2.1.3) πού είναι η logistic συνάρτηση και η σιγμοειδής συνάρτηση, αντίστοιχα.
7. Τέλος, ξεκινά ο υπολογισμός για να βρεθεί το theta που οδηγεί το J(θ) στο τοπικό μέγιστο.

Βρίσκοντας τις τιμές των θ, τις τοποθετούμε στην εξίσωση z και h, έτσι ώστε να υπολογιστούν οι έξοδοι νέων τιμών (που δεν βρίσκονται στο training dataset).

4.2.2. Decision Tree

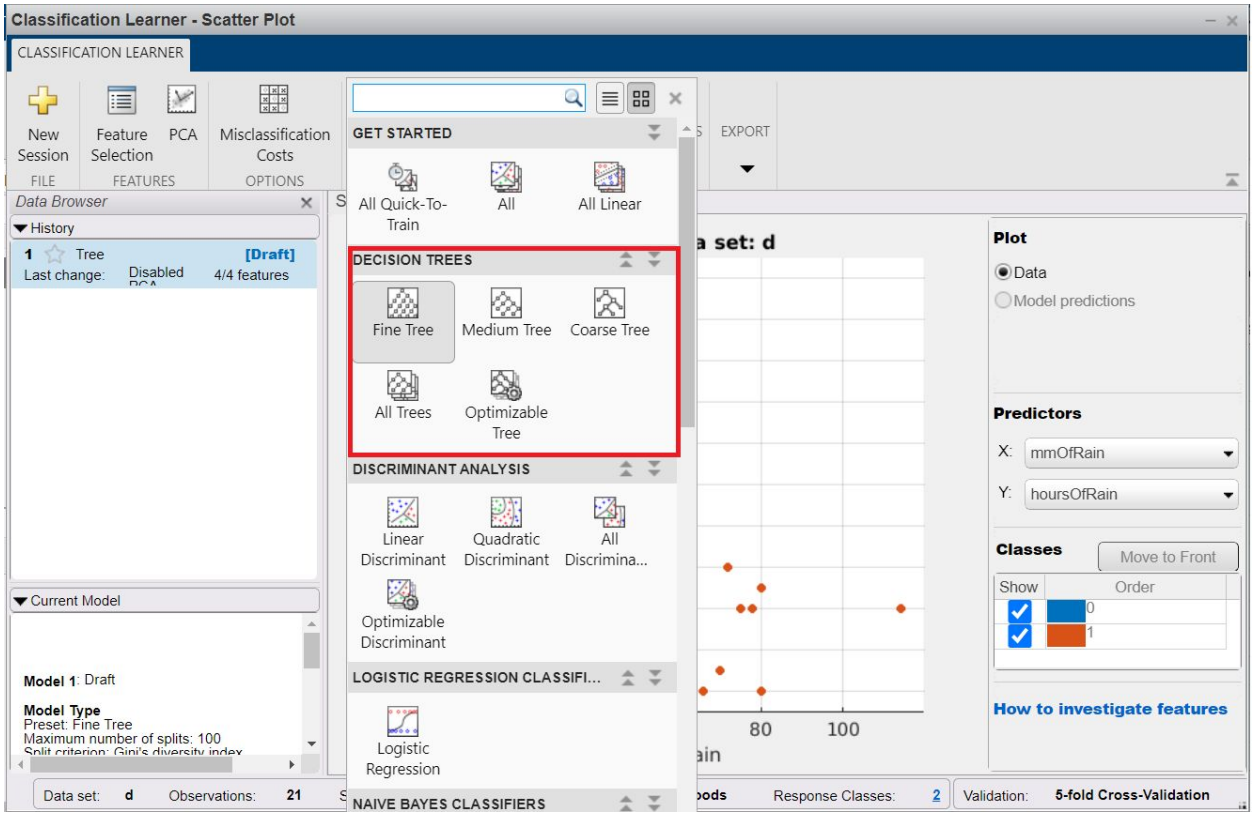
Το Decision Tree έχει την ίδια λογική με ένα λογικό διάγραμμα(flowchart). Είναι αρκετά απλή η λογική του, διότι μπορεί να παρομοιαστεί σαν ερωτήσεις που αν απαντηθούν όλες τότε βγάζουμε το ζητούμενο αποτέλεσμα, για παράδειγμα, για το πρόβλημα που αναφέρθηκε στην παρούσα διπλωματική, το decision tree θα μπορούσε να έχει την παρακάτω μορφή.

Ο κάθε κόμβος αντιπροσωπεύει ένα ερώτημα (κουτάκια χωρίς χρώμα), κάθε σύνδεσμος αντιπροσωπεύει έναν κανόνα και κάθε φύλλο(κουτάκια με χρώμα) αντιπροσωπεύει ένα αποτέλεσμα.



Εικόνα 4.5: Decision Tree

Στο Matlab, θα ήταν πολύ εύκολο να δημιουργηθεί το μοντέλο αυτό με την χρήση του Classification Learner.

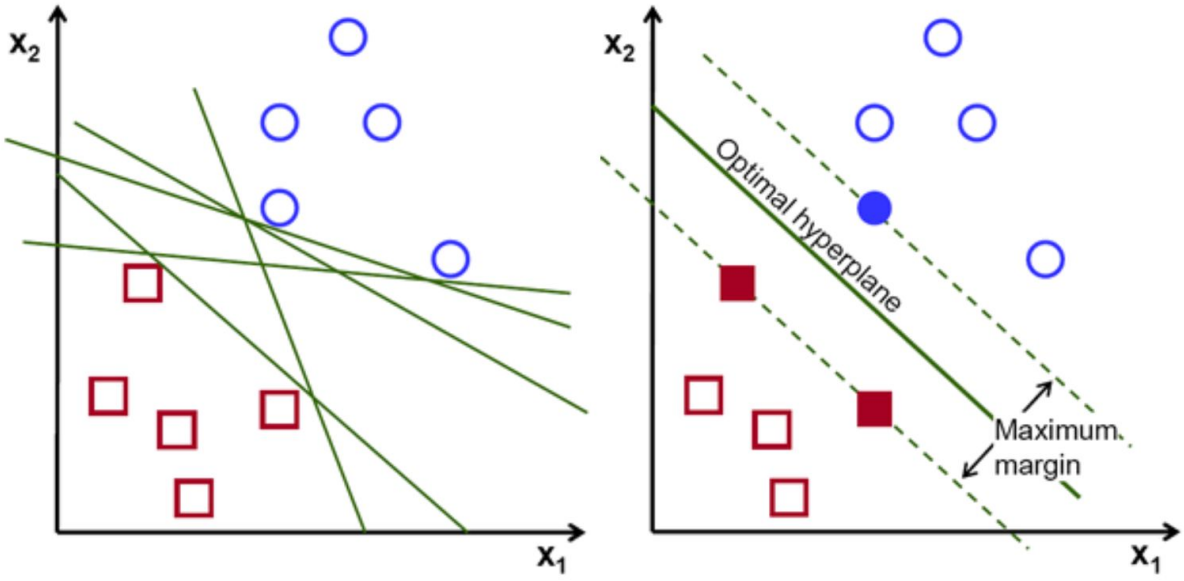


Εικόνα 4.6: Classification Learner (πρόγραμμα Matlab)

4.2.3. Support Vector Machines

Ο αλγόριθμος SVM είναι ένας από τους πιο αποτελεσματικούς ταξινομητές για την ταξινόμηση δεδομένων.

Ο στόχος του αλγορίθμου είναι η εύρεση μιας ευθείας, καμπύλης ή γραφικής παράστασης σε έναν N-διάστατο χώρο, όπου N ο αριθμός των data/μεταβλητών του dataset. Αυτή η γραφική πρέπει να ταξινομεί/χωρίζει όλα τα δεδομένα, όπως φαίνεται στην εικόνα 4.7. Για να διαχωριστούν τα δύο classes (0 και 1), υπάρχουν πολλές πιθανές γραφικές που θα μπορούσαν να επιλεγούν. Πρέπει όμως να βρεθεί μία γραφική που έχει το μέγιστο περιθώριο, δηλαδή τη μέγιστη απόσταση μεταξύ των σημείων δεδομένων και των δύο κατηγοριών, ώστε σε περίπτωση μελλοντικών δεδομένων να υπάρχει αρκετό περιθώριο να μην πέσουν σε άλλη κατηγορία. Όπως φαίνεται και στην παρακάτω εικόνα, υπάρχουν πολλές πιθανές γραφικές άλλα επιλέχθηκε αυτή που έχει το μεγαλύτερο περιθώριο(margin) μεταξύ του class 1 και classs 2.



Εικόνα 4.7: Παράδειγμα Support Vector Machine (Πηγή - Ιστοσελίδα: https://towardsdatascience.com/svm-feature-selection-and-kernels-840781cc1a6c)

Όπως είδαμε και στο κεφάλαιο 4.2.1, τα data που έχουμε είναι 4, συνεπώς θα έχουμε 4-δισδιάστατο χώρο πράγμα που είναι δύσκολο να λυθεί θεωρητικά.

Στο Matlab, θα ήταν πολύ εύκολο να δημιουργηθεί το μοντέλο αυτό με την χρήση του Classification Learner.

4.3. Επιλογή αλγορίθμου

Το τελευταίο βήμα είναι, αφού ολοκληρωθεί η διαδικασία της εκπαίδευσης, είναι να επιλεχθεί ο αλγόριθμός που επιστρέφει την μεγαλύτερη ακρίβεια (δηλαδή το μικρότερο σφάλμα). Οι αλγόριθμοι που επιλέχθηκαν χρησιμοποιούνται για διαφορετικές εφαρμογές, όπως για παράδειγμα ο αλγόριθμος logistic regression λύνει προβλήματα όπως εντοπισμό καρκίνου, για μάρκετινγκ (π.χ. Ο καταναλωτής θα αγοράσει ένα προϊόν ή όχι) και για αποτέλεσμα, όπως πρόβλεψη αν θα μια ομάδα ποδοσφαίρου θα κερδίσει ή όχι. Ο αλγόριθμος SVM χρησιμοποιείται κυρίως για ταξινόμηση εικόνων, αναγνώριση χειρόγραφων κειμένων και για εντοπισμό καρκίνου. Το decision tree είναι μη-γραμμικό που σημαίνει ότι μπορεί να δεχτεί πολλά δεδομένα και να προβλέψει πολλά πιθανά αποτελέσματα.

- Ο αλγόριθμος SVM λειτουργεί πολύ καλά με δεδομένα όπως εικόνες και γράμματα. Το logistic regression λειτουργεί καλύτερα με αναγνωρισμένες ανεξάρτητες μεταβλητές ενώ το decision tree δουλεύει πολύ καλά με γράμματα αλλά και ανεξάρτητες μεταβλητές

- Ο αλγόριθμος SVM βασίζεται στις γεωμετρικές ιδιότητες των δεδομένων. Το logistic regression βασίζεται σε στατιστικές προσεγγίσεις ενώ το decision tree βασίζεται στην πιθανότητα να συμβεί ένα γεγονός ή όχι σε συναρτήσει με τα δεδομένα του dataset.

Συνεπώς αν κρίνουμε τα δεδομένα που θα αποτελούν το dataset (χιλιοστά και οι ώρες βροχής, τα γεωμορφολογικά στοιχεία και η κατάσταση των φρεατίων), ο κατάλληλος αλγόριθμος είναι το logistic regression.
Θα μπορέσουμε, όμως, να κρίνουμε με ακρίβεια το κατάλληλο αλγόριθμο αφού δοκιμάσουμε και τους τρεις στην πράξη. Στα πλαίσια της διπλωματικής δεν μπορούμε να το δούμε πρακτικά, διότι δεν έχουμε τα απαραίτητα δεδομένα.

4.4. Επιπλέον σενάρια

Εκτός από το αν θα πραγματοποιηθεί πλημμύρα ή όχι, μπορούν να εξαχθούν κι άλλες σημαντικές πληροφορίες όπως:

Σενάριο 1: Μετά την φραγή ενός φρεατίου, σε πόσα λεπτά θα φραγούν τα επόμενα φρεάτια;
Έστω ότι σε μια περιοχή τοποθετηθούν οι συσκευές ανίχνευσης φραγής σε ορισμένα φρεάτια, τα οποία βρίσκονται σε κοντινή απόσταση μεταξύ τους. Όπως αναφέρθηκε στο υποκεφάλαιο 2.1, κατά την διάρκεια μιας βροχής, οι σωλήνες του φρεατίου φράζουν, διότι συσσωρεύονται πολύ νερό στους κεντρικούς σωλήνες. Συλλέγοντας μετρήσεις από την συσκευή, μπορούμε να βρούμε το πρώτο φρεάτιο στο οποίο θα ανιχνευτεί φραγή. Στην συνέχεια, βρίσκουμε το δεύτερο κ.ο.κ. Ο στόχος είναι να βρεθεί ένα επαναλαμβανόμενο μοτίβο. Εφόσον βρεθεί ένα μοτίβο, οι δήμοι θα ξέρουν ότι αν ανιχνευτεί φραγή σε έναν Χ αριθμό φρεατίων, τότε θα πρέπει να βρίσκονται σε επιφυλακή.

Σενάριο 2: Αν παρατηρείται ότι ένα φρεάτιο πλημμυρίσε σε λιγότερη ώρα από ότι συνήθως, τότε τι το προκάλεσε;
Αυτό το σενάριο έχει άμεση συσχέτιση με το σενάριο 1. Έστω ότι έχουν καταγραφεί οι ώρες που φράζουν οι σωλήνες των φρεατίων (ανάλογα με τα χιλιοστά βροχής). Σε περίπτωση που παρατηρηθεί ότι ένα συγκεκριμένο φρεάτιο βούλωσε σε λιγότερο χρόνο από ότι συνήθως, τότε σημαίνει ότι επιδράσε ενας εξωτερικός παράγοντας. Συνεπώς μπορεί να πραγματοποιηθουν έρευνες στο συγκεκριμένο φρεάτιο, για να βρεθεί τι προκάλεσε την φραγή του.

Σενάριο 3: Τα φρεάτια που μετά από λίγα χιλιοστά βροχής παρουσιάζουν φραγή, μπορούν να καταγραφούν και να ανακατασκευαστούν.
Σε περιπτώσεις βροχής με πολύ λίγα χιλιοστά νερού, κανένα φρεάτιο δεν θα πρέπει να παρουσιάσει πρόβλημα. Αν ανιχνευθεί φραγή σε κάποιο φρεάτιο, τότε σημαίνει είτε οτι επιδρά

ένας εξωτερικός παράγοντας ή ότι το φρεάτια είναι ελαττωματικό. Συνεπώς, έχοντας αυτά τα δεδομένα, μπορούν να καταγραφούν τα ελαττωματικά φρεάτια και να ανακατασκευαστούν.

ΚΕΦΑΛΑΙΟ 5: ΤΟΠΟΘΕΤΗΣΗ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΕΞΕΛΙΞΕΙΣ

Όπως ειπώθηκε στο κεφάλαιο 2.1, υπάρχουν πολλά είδη φρεατίων. Η συγκεκριμένη διπλωματική θα ασχοληθεί με τα τετραγωνισμένα φρεάτια Για να τοποθετηθεί μέσα στο φρεάτιο, πρέπει να δούμε την συσκευασία στην οποία θα εισέλθει και πως ακριβώς θα μπει του φρεατίου.. Σε αυτό το κεφάλαιο θα ασχοληθούμε με αυτό. Επίσης, σε αυτό το κεφάλαιο θα αναφέρουμε μελλοντικές εξελίξεις της συσκευής.

5.1 Συσκευασία

Η συσκευασία στην οποία θα τοποθετηθεί η συσκευή πρέπει οπωσδήποτε να έχει ορισμένα χαρακτηριστικά, όπως

- Σκληρό υλικό. Πρέπει να είναι αρκετά σκληρό ώστε να μην σπάσει σε περίπτωση εισέρχεται νερό στο φρεάτιο, με μεγάλη ορμή.
- Εύκαμπτο υλικό. Η συσκευασία θα πρέπει να βιδωθεί στον τοίχο του φρεατίου, συνεπώς πρέπει να είναι εύκαμπτο υλικό το οποίο θα μπορεί να δεχθεί τις δυνάμεις από τα καρφιά και επίσης να μπορεί να πάρει μία κλίση, χωρίς να σπάσει.
- Αδιάβροχο. Το Arduino,οι μπαταρίες, το DHT11 και το ESP8266-01 είναι στοιχεία τα οποία δεν πρέπει να έρχονται σε επαφή με το νερό. Συνεπώς η συσκευασία πρέπει να είναι αεροστέγης, έτσι ώστε ακόμα και σε περίπτωση πλημμύρας να μην εισχωρήσει νερό.
- Να περιέχει εσοχές. Θα πρέπει να περιέχει εσοχές έτσι ώστε να το Floater κι το Water Level Sensor να συνδέονται μεν με το arduino, αλλα να βρίσκονται έξω από την συσκευασία.
- Κι τέλος, να είναι εύκολη η αποσυναρμολόγηση για περιπτώσεις αλλαγής μπαταρίας ή σφάλματος των αισθητηρίων.

Το υλικό που ταιριάζει σε όλα τα παραπάνω bullets είναι το πλαστικό, όπως το ABS το οποίο είναι σκληρό αλλά κι εύκαμπτο παράλληλα.

Εικόνα 5.1: Πλαστικο κουτί 90mmX90mm (πηγή - ιστοσελίδα: https://www.kafkas.gr/ilektrologiko-yliko/plastika-spiral-koutia-chalyvdina/vareos-typou/koutia-vareos-typou-betou/koutia-vareos-typou/obo-bettermann-kouti-epitoicho-7eis.90mmx90mm-ip55-gkri-tetragono_197383/):

Μέσα στο κουτί θα είναι τοποθετημένα η μπαταριοθήκη κι από πάνω του το Arduino. Το αισθητήριο υγρασίας DHT11 και το ESP wi-fi module θα βρίσκονται εντός του κουτιού. Έξω από το κουτί βρίσκονται αισθητήρας στάθμης νερού και ο αισθητήρας επίπλευσης. Θα πρέπει έπειτα να κλείσουν όλες οι εσοχές ωστέ το νερό να μην μπορεί να εισέλθει μέσα στο κουτί.

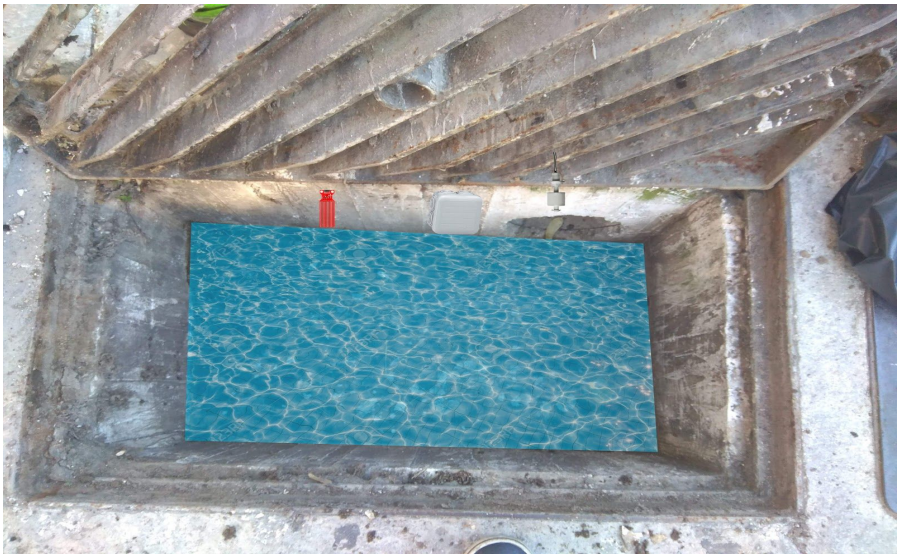
5.2 Τοποθέτηση

Η τοποθέτηση μέσα στο φρεάτιο θα είναι όσο πιο απλή γίνεται ώστε να μπορεί να γίνει εύκολα αποσυναρμολόγηση σε περιπτώσεις αλλαγής μπαταρίας ή σφάλματος αισθητηρίων. Με την λογική που έχει υλοποιηθεί η ιδέα αυτή, η τοποθέτηση θα πρέπει να γίνει ως εξής:

- Η συσκευασία που θα περιέχει το Arduino, μπαταρίες, αισθητήρα DHT11 και ESP8266-01 θα πρέπει να τοποθετηθεί στο ψηλότερο σημείο δυνατόν. Προφανώς δεν μπορεί να τοποθετηθεί ακριβώς κάτω από την σχάρα, συνεπώς, θα πρέπει να μπει λίγα εκατοστά πιο κάτω από αυτήν. Κύριο μέλημα ειναι να τοποθετηθεί με τέτοιο τρόπο ώστε οι υπάλληλοι να μπορούν να καθαρίσουν χωρίς να εμποδίζονται από την συσκευή.
- Το floater θα πρέπει να τοποθετηθεί λίγα εκατοστά πιο πάνω από τον σωλήνα ή στην πάνω μεριά, στο εσωτερικό του σωλήνα υδροσυλλογής. Πρέπει να δούμε, στην πράξη, σε πιο σημείο θα έχουμε τα καλύτερα αποτελέσματα. Για παράδειγμα, σε περίπτωση που το φλοτέρ μπει στην πάνω μεριά του σωλήνα, τότε αν το νερό εισέρχεται με ορμή στον σωλήνα, τότε το φλοτέρ θα ενεργοποιείται, κάτι το οποίο θα επιστρέφει false positive στην σελίδα. Επίσης αν

το φλοτέρ μπει λίγα εκατοστά πιο πάνω από τον σωλήνα, μπορεί η στάθμη του νερού να μην φτάσει το σημείο όπου βρίσκεται ο αισθητήρας και να μην ανιχνευθεί ποτέ η φραγή. Συνεπώς, με προσοχή, πρέπει να επιλέξουμε το κατάλληλο σημείο.

- Το water senor θα πρέπει να τοποθετηθεί περίπου στην μέση του φρεατίου, λιγο πιο πάνω από τον αισθητήρα, έτσι ώστε να γνωρίζουμε μέχρι περίπου πιο σημείο είναι πλημμυρισμένο το φρεάτιο.



Εικόνα 5.2: Τοποθέτηση συσκευής εντός φρεατίου

Σίγουρα, στην τοποθέτηση θα προκληθούν ορισμένα ερωτήματα όπως:
Για την συσκευή:

- Η συσκευή μπορεί να αντέξει σε περίπτωση μεγάλης καταιγίδας;

Για το αισθητήριο επίπλευσης:

- Αν πέσει ένα σκουπίδι πάνω, τότε πως θα μπορέσει ο αισθητήρας να λειτουργήσει σωστά;

Για το αισθητήριο στάθμης νερού:

- Σε περίπτωση που χώμα κολλήσει πάνω στον αισθητήρα, θα δείχνει λάθος μετρήσεις;

Για την συσκευή πρέπει να δοκιμαστεί κάτω από πίεση ώστε να δούμε αν θα αντέξει κάτω από συνθήκες πλημμύρας. Για το αισθητήριο επίπλευσης, αν πέσει ένα σκουπίδι πάνω του, τότε σε περίπτωση φραγής του φρεατίου, η βαρύτητα θα σηκώσει το σκουπίδι στην επιφάνεια απελευθερώνοντας τον αισθητήρα. Αν δεν γίνει αυτό, τότε ο αισθητήρας στάθμης νερού θα έχει τιμή 500 ενώ ο αισθητήρας επίπλευσης τιμή 0, που δεν είναι δυνατόν, συνεπώς θα γνωρίζουμε ότι υπάρχει ένα σφάλμα. Αντίστοιχα για το αισθητήρα στάθμης, εφόσον δεν βρέχει κι ο αισθητήρας επίπλευσης έχει κολλήσει σε μία τιμή, σημαίνει ότι υπάρχει κάποιο σφάλμα.

5.3 Μελλοντικές Εξελίξεις

Σε περίπτωση μαζικής παραγωγής οι εξελίξεις θα μπορούσαν να χωριστούν σε τρεις τομείς.

1.Εξελίξεις όσον αφορά την συσκευή:

- Χρήση αισθητηρίων που προσφέρουν μεγαλύτερη ακρίβεια κι έχουν μεγαλύτερο προσδόκιμο ζωής
- Αλλαγή μικροελεγκτή σε arduino nano ή, ακόμα καλύτερα, δημιουργία πλακέτας έτσι ώστε να είναι πιο compact η συσκευή

2. Εξελίξεις όσον αφορά την ιστοσελίδα:

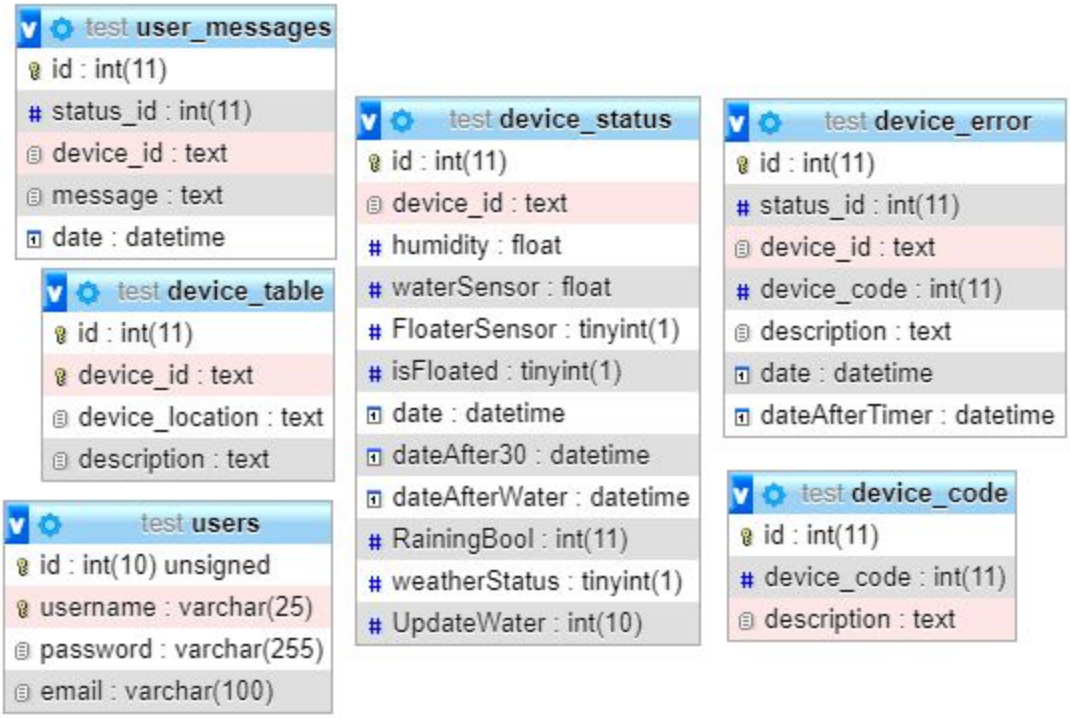
- Δημιουργία POST API που θα επιτρέπει την εύκολη εγγραφή μια νέας συσκευής
- Αλλαγή της λογικής του χάρτη ώστε να μπορεί να υποστηρίξει πολλές συσκευές
- Προσθήκη for loop στην σελίδα ώστε να υποστηρίζει περισσότερες από μία συσκευή

3. Εξελίξεις όσον αφορά το Machine Learning αλγόριθμο

- Σε αυτό το κομμάτι, σημαντικό είναι να συλλεχθούν δεδομένα από τις συσκευές. Όταν επιτευχθεί αυτό, τότε μπορεί να δημιουργηθεί ο αλγόριθμος ταξινόμησης.

ΠΑΡΑΡΤΗΜΑ Α: ΠΙΝΑΚΕΣ ΣΤΗΝ ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

Στο παράρτημα αυτό θα αναφερθούν όλοι οι πίνακες στην βάση δεδομένων που χρησιμοποιήθηκαν στον σχεδιασμό της σελίδας.
Στην παρακάτω εικόνα παρουσιάζονται όλοι οι πίνακες της βάσης δεδομένων



Εικόνα Α.1: Πίνακες στην βάση δεδομένων

Ο πίνακας device_code περιέχει τους δύο τύπους των error που καταγράφονται από την συσκευή:

id	device_code	description
1	50	Floater is blocked
2	100	Water Level is at maximum level

Εικόνα Α.2 : Πίνακας device_code

Αποτελείται από τις στήλες:

- Id, το οποίο είναι ένα πεδίο που αυξάνεται κάθε φορά που εισέρχεται νέο δεδομένο στο πίνακα
- Device_code. Κάθε σφάλμα έχει δικό του κωδικό. Αν το φλοτέρ πάρει την τιμή 1, τότε ο κωδικός του σφάλματος θα ειναι 50 ενώ αν το αισθητήριο στάθμης έχει τιμή πάνω από 500, τότε ο κωδικός σφάλματος θα ειναι 100.

- Description, ειναι η περιγραφή του σφάλματος.

Στον πίνακα device_status αποθηκεύονται όλες οι μετρήσεις

id ▲ 1	device_id	humidity	waterSensor	FloaterSensor	isFloated	date	dateAfter30	dateAfterWater	RainingBool	weatherStatus	UpdateWater
7	imei24	3	24	45	0	2020-11-15 16:32:04	0000-00-00 00:00:00	2020-01-13 01:14:51	0	0	0
11	imei24	13	12	0	0	2020-12-01 22:08:07	0000-00-00 00:00:00	2020-01-13 01:14:51	0	0	0
12	imei24	13	12	0	0	2020-12-01 22:09:40	0000-00-00 00:00:00	2020-01-13 01:14:51	0	0	0

Εικόνα Α.3 : Πίνακας device_status

Αποτελείται από τις στήλες:

- Id, το οποίο είναι ένα πεδίο που αυξάνεται κάθε φορά που εισέρχεται νέο δεδομένο στο πίνακα
- device_id. Είναι το id/ ονομασία της συσκευής
- Humidity, όπου αποθηκεύονται οι μετρήσεις του αισθητήρα υγρασίας
- waterSensor, όπου αποθηκεύονται οι μετρήσεις του αισθητήρα στάθμης νερού
- FloaterSensor, όπου αποθηκεύονται οι μετρήσεις του αισθητήρα επίπλευσης
- isFloated, όπου αν μετά από 30 λεπτά βροχής ο αισθητήρας επίπλευσης έχει ακόμα την τιμή 1, τότε το isFloated παίρνει την τιμή 1.
- Date, όπου είναι η ημερομηνία που πραγματοποιήθηκε η μέτρηση
- dateAfter30. Μετά το τέλος της βροχής, παίρνουμε την τωρινή ώρα κι προσθέτουμε 30 λεπτά. Αυτό αποθηκεύεται σε αυτήν την στήλη. Αν περάσουν τα 30 λεπτά κι ο αισθητήρας επίπλευσης έχει ακόμα την τιμή 1, τότε το isFloated γίνεται 1.
- dateAfteWater. Αν ο αισθητήρας στάθμης νερού έχει τιμή πάνω από 500, παίρνουμε την τωρινή ώρα κι προσθέτουμε 20 λεπτά. Αυτό αποθηκεύεται σε αυτήν την στήλη. Αν περάσουν τα 20 λεπτά κι ο αισθητήρας έχει ακόμα την τιμή 500, τότε αποθηκεύεται στον πινακα device_error
- RainingBool. Παίρνει δύο τιμές, 0 και 1. Το 0 σημαίνει ότι δεν βρέχει ένω το 1 σημαίνει ότι βρέχει. Χρησιμοποιείται σαν flag.
- weatherStatus, αποθεύεται η κατάσταση του καιρού, δηλαδή αν βρέχει ή όχι.
- updateWater, όπου χρησιμοποιείται για τον αισθητήρα στάθμης. Όταν η τιμή είναι 1, τότε ενεργοποιείται το timer.

Στον πίνακα device_error αποθηκεύονται όλα τα σφάλματα των συσκευών:

id	status_id	device_id	device_code	description	date
71	36	imei24	50	Floater is blocked	2020-12-09 01:30:16
72	36	imei24	50	Floater is blocked	2020-12-09 19:34:09
73	36	imei24	50	Floater is blocked	2020-12-09 19:47:17

Εικόνα Α.4 : Πίνακας device_error

Αποτελείται από τις στήλες:

- Id, το οποίο είναι ένα πεδίο που αυξάνεται κάθε φορά που εισέρχεται νέο δεδομένο στο πίνακα
- Status_id. Παίρνει το id από τον πίνακα device_status, ώστε να γνωρίζουμε σε ποια μέτρηση δημιουργήθηκε το σφάλμα. Αυτοί οι δύο πίνακες έχουν μία συσχέτιση μεταξύ τους.
- Device_code. Κάθε σφάλμα έχει δικό του κωδικό. Αν το φλοτέρ πάρει την τιμή 1, τότε ο κωδικός του σφάλματος θα ειναι 50 ενώ αν το αισθητήριο στάθμης έχει τιμή πάνω από 500, τότε ο κωδικός σφάλματος θα ειναι 100.
- Description, ειναι η περιγραφή του σφάλματος.
- Date. Είναι η ημερομηνία που δημιουργήθηκε το σφάλμα.

Στον πίνακα device_table αποθηκεύονται όλες οι συσκευές

id	device_id	device_location	description
1	imei24	Aigaleo 45 & 50	This freatio is 30m

Εικόνα Α.5 : Πίνακας device_table

Για παράδειγμα,

- Id, το οποίο είναι ένα πεδίο που αυξάνεται κάθε φορά που εισέρχεται νέο δεδομένο στο πίνακα
- Device_id. Είναι το id/ ονομασία της συσκευής
- Device_location. Ειναι η τοποθεσία που έχει τοποθετηθεί η συσκευή
- Description, ειναι η περιγραφή του φρεατίου.

Στον πίνακα users αποθηκεύονται τα διαπιστευτήρια των χρηστών.

id	username	password	email
1	paspartou	paspartou	elenikay2000@yahoo.com

Εικόνα Α.6 : Πίνακας users

Αποτελείται από τις στήλες

- Id, το οποίο είναι ένα πεδίο που αυξάνεται κάθε φορά που εισέρχεται νέο δεδομένο στο πίνακα
- Username, που είναι το όνομα χρήστη
- Password, είναι ο κωδικός του χρήστη (Μελλοντική εξέλιξη: Πρέπει να το κάνουμε encrypt γιατί αποτελεί προσωπικό δεδομένο)

- Email, ειναι το email του χρήστη

Στον πίνακα users_messages αποθηκεύονται δυσλειτουργίες της συσκευής.

id	status_id	device_id	message	date
1	81	imei24	Level of water is high, but floater is 0. Please C...	2021-02-25 00:58:22
2	81	imei24	Level of water is high, but floater is 0. Please C...	2021-02-25 01:12:09
3	81	imei24	No data was registered during rain. Please Check.	2021-02-25 02:47:34

Εικόνα Α.7 : Πίνακας users_messages

Αποτελείται από τις στήλες

- Id, το οποίο είναι ένα πεδίο που αυξάνεται κάθε φορά που εισέρχεται νέο δεδομένο στο πίνακα
- Status_id. Παίρνει το id από τον πίνακα device_status, ώστε να γνωρίζουμε σε ποια μέτρηση δημιουργήθηκε το σφάλμα. Αυτοί οι δύο πίνακες έχουν μία συσχέτιση μεταξύ τους.
- Device_id. Είναι το id/ ονομασία της συσκευής
- Message, όπου είναι η περιγραφή της δυσλειτουργίας.
- Date, ειναι η ημερομηνία που πραγματοποιήθηκε η δυσλειτουργία.

ΠΑΡΑΡΤΗΜΑ Β: APIs

1.db_conn.php

```php
<?php
session_start();
$conn = mysqli_connect("localhost","root","","test",3306);

// Check connection
if ($conn -> connect_errno) {
  echo "Failed to connect to MySQL: " . $conn -> connect_error;
  exit();
}
```

Εικόνα Β.1 : Κώδικας PHP σύνδεσης στην βάση δεδομένων

Εδώ πραγματοποιείται η σύνδεση με το database. Η εντολή που πραγματοποιεί την σύνδεση είναι το mysql_connect, με τις μεταβλητές του localhost, του username, του password, του database και του port. Σε περίπτωση που δεν συνδεθεί με τη βάση, επιστρέφει μήνυμα λάθους.

HTTP GET REQUESTS

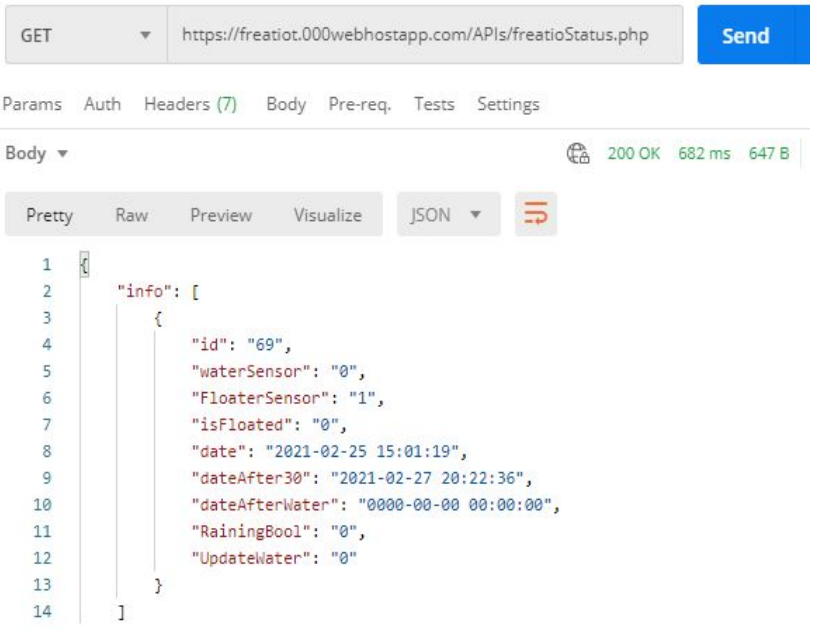
1.freatioStatus.php

```php
<?php
    include_once('db_conn.php');
    $task = isset($_GET['task']) ? mysqli_real_escape_string($conn, $_GET['task']) :  "";
    $sql = "SELECT id,waterSensor, FloaterSensor, isFloated,date, dateAfter30,dateAfterWater, RainingBool FROM device_status ORDER BY id DESC
        LIMIT 1";
    $get_data_query = mysqli_query($conn, $sql) or die(mysqli_error($conn));
        if(mysqli_num_rows($get_data_query)!=0){
        $result = array();

        while($r = mysqli_fetch_array($get_data_query)){
            extract($r);
            $result[] = array("id" => $id,  "waterSensor" => $waterSensor, "FloaterSensor" => $FloaterSensor, "isFloated" => $isFloated, "date"
                => $date , "dateAfter30" =>$dateAfter30,"dateAfterWater" =>$dateAfterWater, "RainingBool" =>$RainingBool);
        }
        $json = array("info" => $result);
    }
    else{
        $json = array("status" => 0, "error" => "To-Do not found!");
    }
    @mysqli_close($conn);
    // Set Content-type to JSON
    header('Content-type: application/json');
    echo json_encode($json);
?>
```

Εικόνα Β.2 : Κώδικας PHP freatioStatus.php

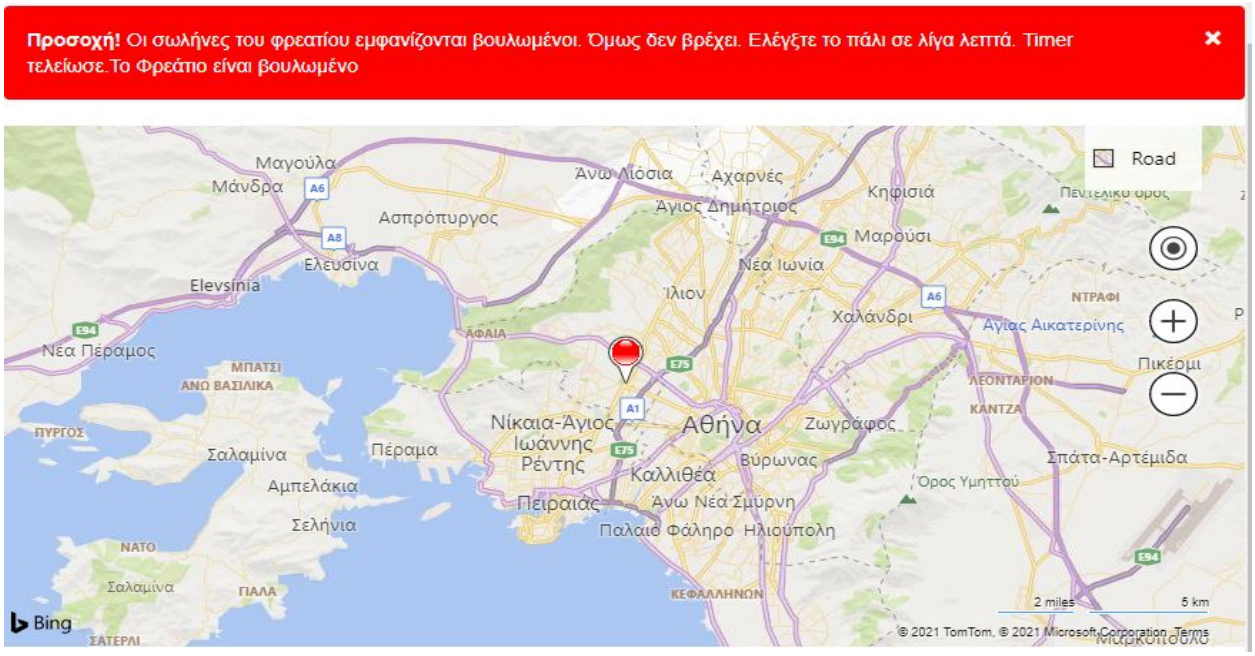
API: GET https://freatiot.000webhostapp.com/APIs/freatioStatus.php

Αυτό το API επιστρέφει τις μετρήσεις της συσκευής. Χρησιμοποιώντας το πρόγραμμα Postman μπορούμε να δούμε τι πληροφορίες παίρνουμε με αυτό το API.



Εικόνα B.3 : API HTTP GET freatioStatus.php (πρόγραμμα Postman)

Αυτό το API έχει χρησιμοποιηθεί σε διάφορα σημεία του κώδικα όπως σε όλα error messages που εμφανίζονται στην σελίδα, όπως για παράδειγμα η παρακάτω εικόνα. Επίσης, έχει χρησιμοποιηθεί στο countdown των 30 λεπτών.



Εικόνα B.4 : Μήνυμα σφάλματος

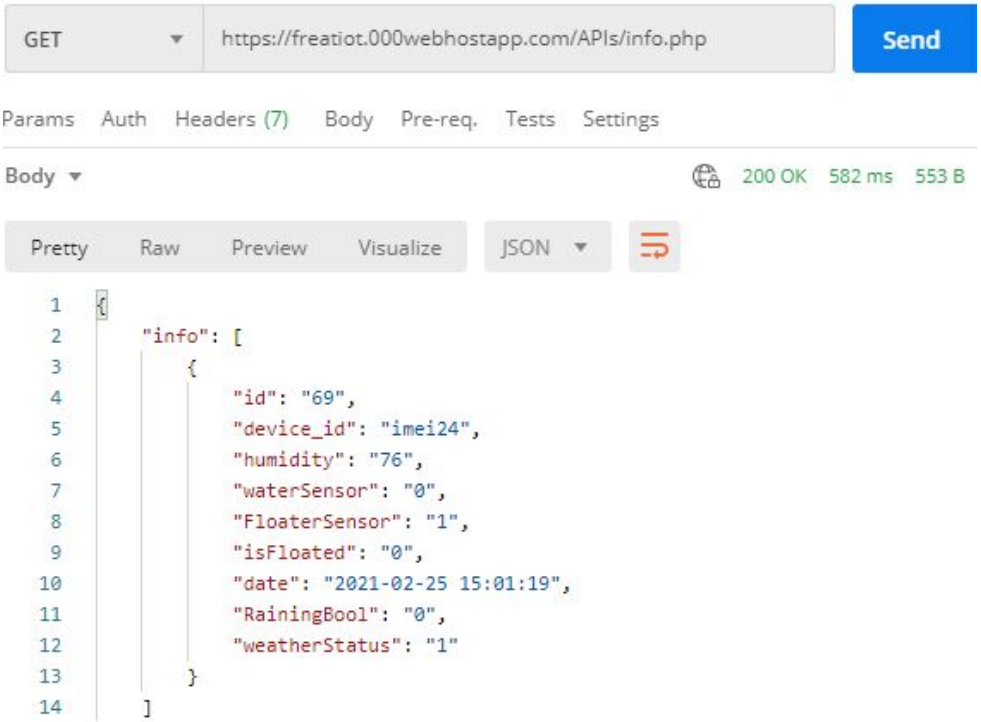
2.Info.php

```php
<?php
//API: GET INFO FOR FREATIO
    include_once('db_conn.php');
    $task = isset($_GET['task']) ? mysqli_real_escape_string($conn, $_GET['task']) :  "";
    $sql = "SELECT * FROM device_status ORDER BY id DESC LIMIT 1"; //SELECT * FROM `device_error`order by id DESC LIMIT 10
    $get_data_query = mysqli_query($conn, $sql) or die(mysqli_error($conn));
        if(mysqli_num_rows($get_data_query)!=0){
        $result = array();

        while($r = mysqli_fetch_array($get_data_query)){
            extract($r);
            $result[] = array("id" => $id, "device_id" => $device_id, 'humidity' => $humidity,'waterSensor' => $waterSensor,'FloaterSensor' =>
                $FloaterSensor,"isFloated" => $isFloated, "date" => $date, "RainingBool" => $RainingBool, "weatherStatus" => $weatherStatus )
                ;
        }
        $json = array("info" => $result);
    }
    else{
        $json = array("status" => 0, "error" => "To-Do not found!");
    }
@mysqli_close($conn);
// Set Content-type to JSON
header('Content-type: application/json');
echo json_encode($json);
?>
```

Εικόνα Β.5 :Κώδικας PHP info.php

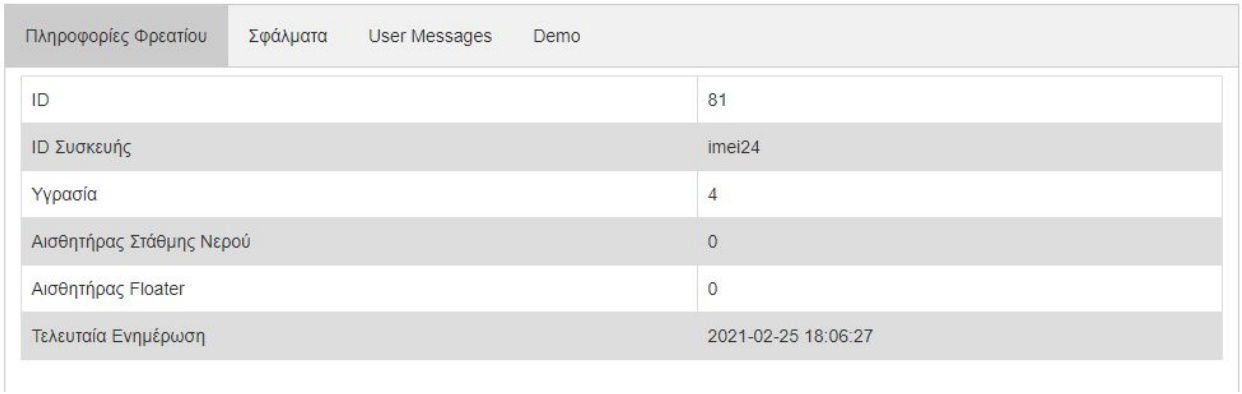
API: GET https://freatiot.000webhostapp.com/APIs/info.php

Αυτό εδώ το API είναι σχεδόν ακριβώς ίδιο με το freatioStatus.php που είδαμε προηγουμένως, απλά έχει μικροδιαφορές.



Εικόνα Β.6. : API HTTP GET info.php (πρόγραμμα Postman)

Συγκεκριμένα αυτό το API χρησιμοποιείται σε αυτό το σημείο του κώδικα για τις πληροφορίες του πίνακα “Πληροφορίες Φρεατίου”



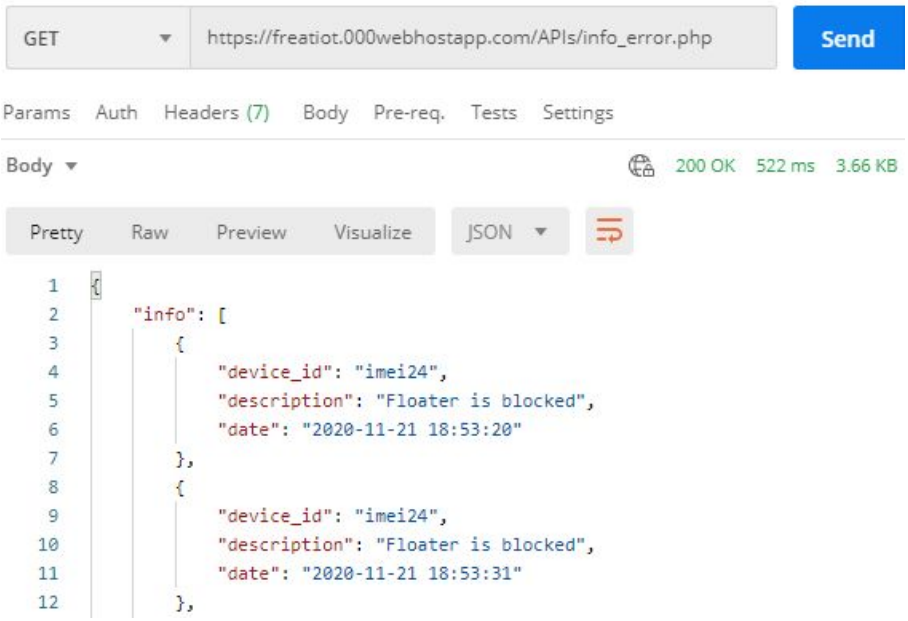
Εικόνα Β.7 : Πίνακας πληροφορίες φρεατίου

3. Info_error.php

```php
<?php
//API: GET error entries
    include_once('db_conn.php');
    $task = isset($_GET['task']) ? mysqli_real_escape_string($conn, $_GET['task']) :  "";
    $sql = "SELECT date,device_id, description FROM device_error ORDER BY id";
    $get_data_query = mysqli_query($conn, $sql) or die(mysqli_error($conn));
        if(mysqli_num_rows($get_data_query)!=0){
        $result = array();

        while($r = mysqli_fetch_array($get_data_query)){
            extract($r);
            $result[] = array('device_id' => $device_id,'description' => $description,"date" => $date );
        }
        $json = array("info" => $result);
    }
    else{
        $json = array("status" => 0, "error" => "To-Do not found!");
    }
@mysqli_close($conn);
// Set Content-type to JSON
header('Content-type: application/json');
echo json_encode($json);
?>
```

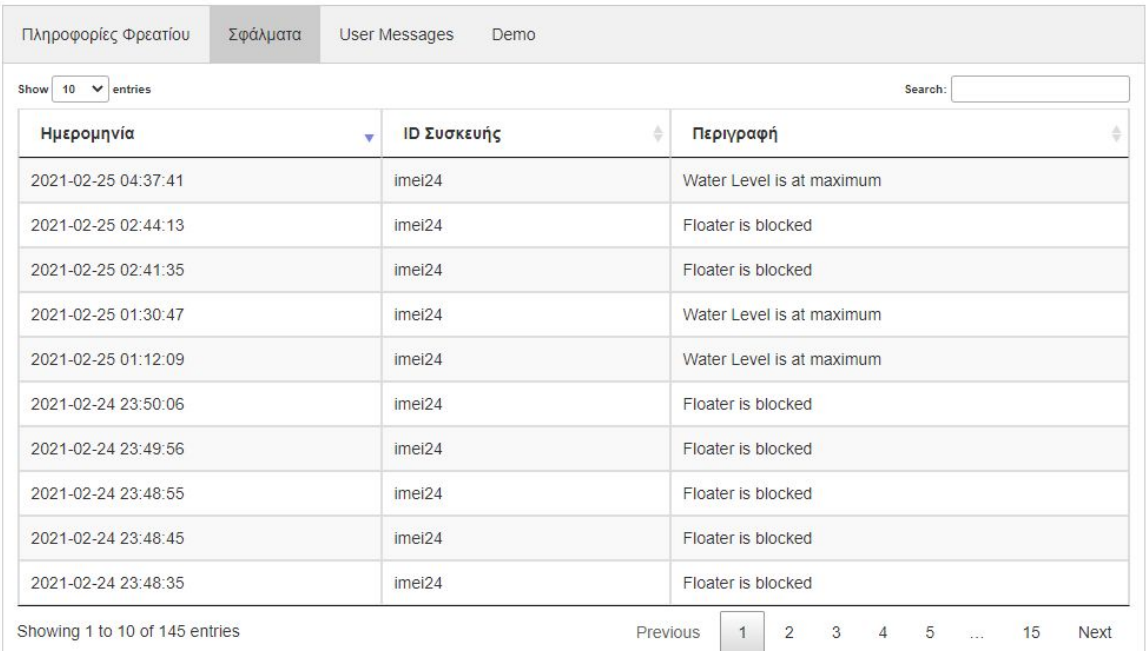
Εικόνα Β.8 :Κώδικας PHP info_error.php

API: GET https://freatiot.000webhostapp.com/APIs/info_error.php



Εικόνα Β.9 : API HTTP GET info_error.php (πρόγραμμα Postman)

Αυτό το API παίρνει απο την βάση, τα errors της συσκευής και εμφανίζεται στο tab "Σφάλματα".



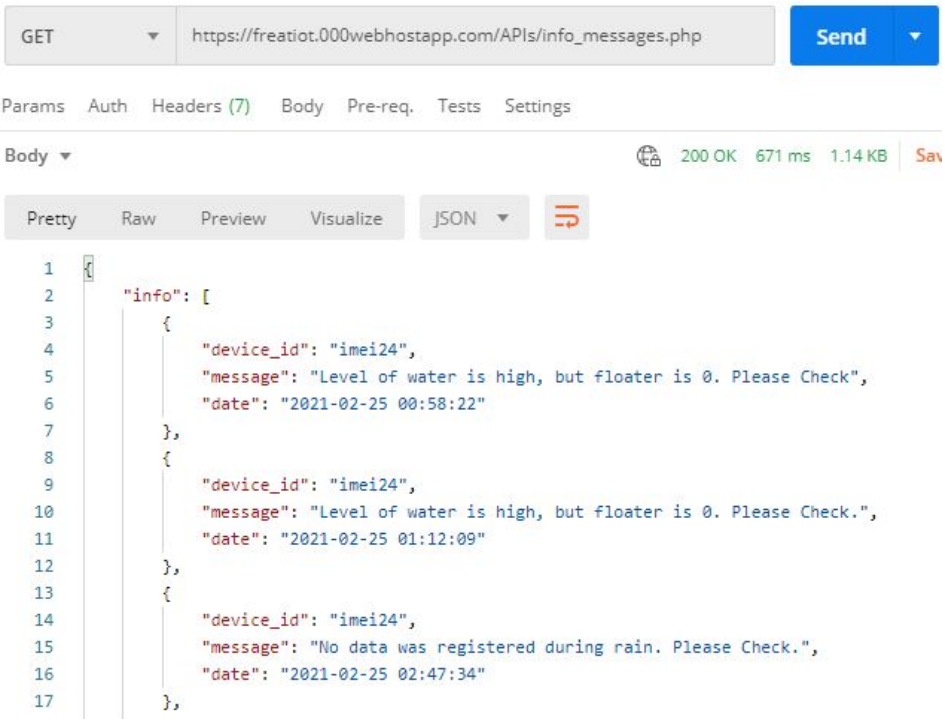
Εικόνα Β.10 : Πίνακας σφάλματα

4. Info_Messages.php

```php
<?php
//API: GET error entries
    include_once('db_conn.php');
    $task = isset($_GET['task']) ? mysqli_real_escape_string($conn, $_GET['task']) :  "";
    $sql = "SELECT device_id, message,date FROM user_messages ORDER BY id";
    $get_data_query = mysqli_query($conn, $sql) or die(mysqli_error($conn));
        if(mysqli_num_rows($get_data_query)!=0){
        $result = array();

        while($r = mysqli_fetch_array($get_data_query)){
            extract($r);
            $result[] = array('device_id' => $device_id,'message' => $message,"date" => $
                date );
        }
        $json = array("info" => $result);
    }
    else{
        $json = array("status" => 0, "error" => "To-Do not found!");
    }
@mysqli_close($conn);
// Set Content-type to JSON
header('Content-type: application/json');
echo json_encode($json);
?>
```

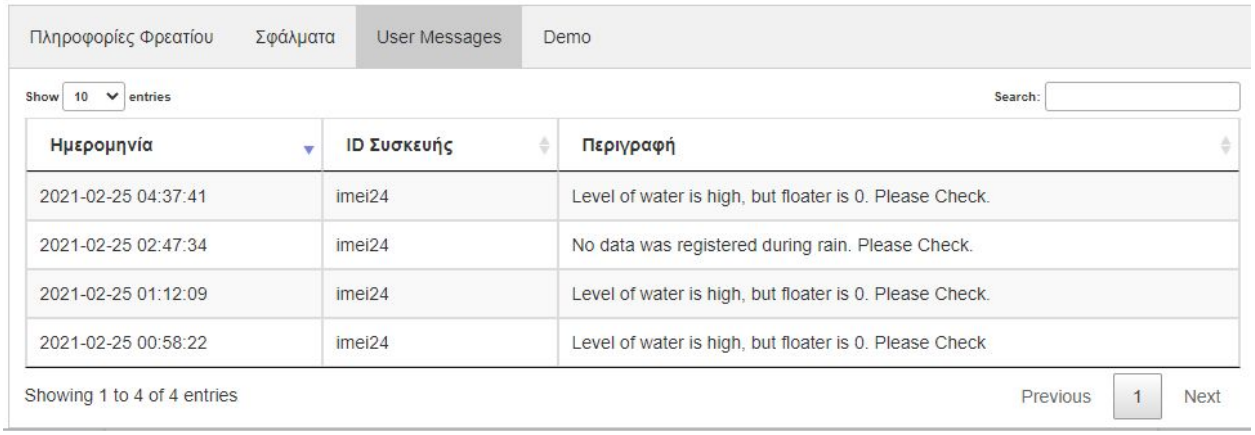
Εικόνα B.11 :Κώδικας PHP Info_Messages.php

API: GET https://freatiot.000webhostapp.com/APIs/info_messages.php



Εικόνα B.12 : API HTTP GET info_messages.php (πρόγραμμα Postman)

Αυτό το API παίρνει απο την βάση, όλα τα user messages, δηλαδή τις δυσλειτουργίες της συσκευής και τις εμφανίζει στην σελίδα, στον πίνακα User Messages.



Εικόνα Β.13 : Πίνακας User Messages

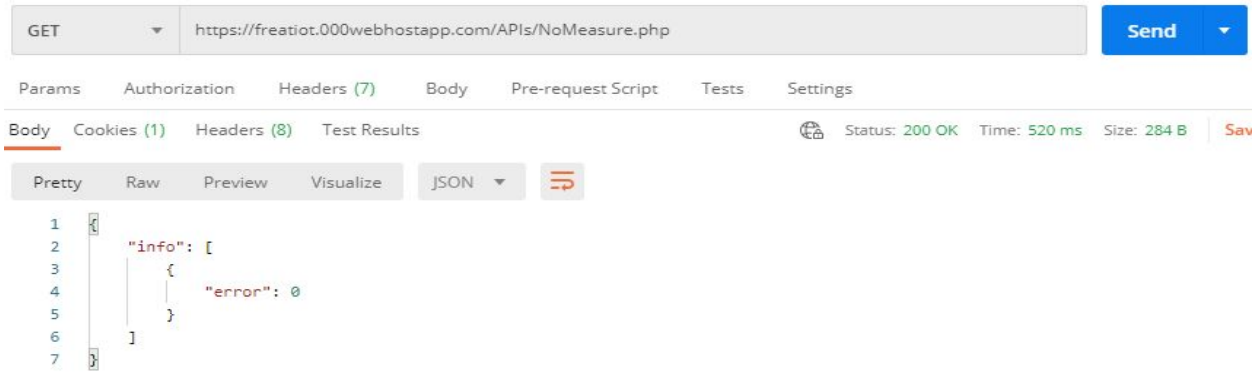
5. Nomeasure.php

```php
<?php
//API: GET error entries
    include_once('db_conn.php');
    $task = isset($_GET['task']) ? mysqli_real_escape_string($conn, $_GET['task']) :  "";
    $sql = "SELECT SUM(waterSensor), SUM(FloaterSensor) FROM
    (SELECT id, waterSensor, FloaterSensor FROM `device_status` ORDER BY id DESC LIMIT 2) t1";
    $get_data_query = mysqli_query($conn, $sql) or die(mysqli_error($conn));

    if(mysqli_num_rows($get_data_query)!=0){
        $result = array();

        while($r = mysqli_fetch_array($get_data_query)){
            extract($r);

        $level = $r[0];
        $float = $r[1];


        if ($level == 0 && $float == 0)
        {
            $error = 1;
        }
        else
        {
            $error = 0;
        }

            $result[] = array('error' => $error);
        }
        $json = array("info" => $result);
    }
    else{
        $json = array("status" => 0, "error" => "To-Do not found!");
    }
@mysqli_close($conn);
// Set Content-type to JSON
header('Content-type: application/json');
echo json_encode($json);
?>
```

Εικόνα Β.14 :Κώδικας PHP NoMeasure.php

API: GET https://freatiot.000webhostapp.com/APIs/NoMeasure.php

Το 2ο user message που στέλνουμε είναι “No data was registered during rain. Please Check.”. Για να γίνει αυτό, παίρνουμε το sum/άθροισμα της στήλης waterSensor και ο sum/άθροισμα της στήλης FloaterSensor για τις τελευταίες 30-50 μετρήσεις (το sql αυτή την στιγμή παίρνει τις δύο τελευταίες τιμές για τα πλαίσια της διπλωματικής αυτής.). Αν το άθροισμα και των δύο είναι 0, υποθέτουμε ότι υπάρχει μια δυσλειτουργία. Το API επιστρέφει την τιμή error = 0, αν το άθροισμα δεν είναι 0 και την τιμή error = 1 αν το άθροισμα είναι 0.



Εικόνα B.15 : API HTTP GET NoMeasure.php (πρόγραμμα Postman)

HTTP POST REQUESTS

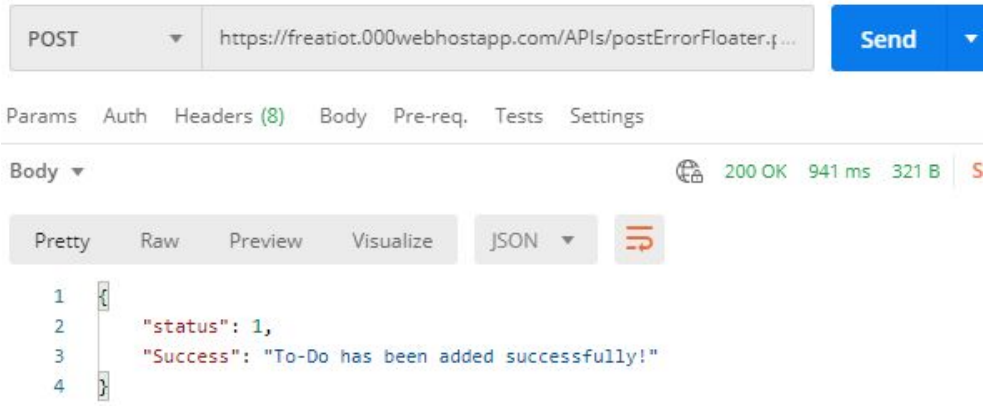
1.postErrorFloater.php

```php
<?php
//POST error to table
    include_once('db_conn.php');
    if($_SERVER['REQUEST_METHOD'] == "POST"){
        // Get data from the REST client
        $task = isset($_POST['task']) ? mysqli_real_escape_string($conn, $_POST['task']) : "";
        $date = isset($_POST['date']) ? mysqli_real_escape_string($conn, $_POST['date']) : "";
        $priority = isset($_POST['priority']) ? mysqli_real_escape_string($conn, $_POST['priority']) : "";

        $id;
        $device_id;
        $device_code = 50;
        $description = 'Floater is blocked';

        //Get values from another table
        $sql1 = "SELECT * FROM device_status ORDER BY id DESC LIMIT 1";
        $get_data_query = mysqli_query($conn, $sql1) or die(mysqli_error($conn));
            if(mysqli_num_rows($get_data_query)!=0){

            if (mysqli_num_rows($get_data_query) > 0) {
              while($row = mysqli_fetch_assoc($get_data_query)) {
               $id = $row["id"];
               $device_id =  $row["device_id"];
              }
            }
        }

    // Insert data into database
    $sql = "INSERT INTO `device_error`(`status_id`, `device_id`, `device_code`, `description`, `date`) VALUES ('.$id.','$device_id',50,'Floater is blocked',now())";


    $post_data_query = mysqli_query($conn, $sql);
    if($post_data_query){
        $json = array("status" => 1, "Success" => "To-Do has been added successfully!");
    }
    else{
        $json = array("status" => 0, "Error" => "Error adding To-Do! Please try again!");
    }
}
else{
    $json = array("status" => 0, "Info" => "Request method not accepted!");
}
@mysqli_close($conn);
// Set Content-type to JSON
header('Content-type: application/json');
echo json_encode($json);

?>
```

Εικόνα Β.16: Κώδικας PHP postErrorFloater.php

API: POST https://freatiot.000webhostapp.com/APIs/postErrorFloater.php

Το API αυτό χρησιμοποιείται για να γραφτεί στην βάση την στιγμή που πραγματοποιηθεί ένα error. Μέσω Postman, το response του POST API είναι “ To-Do has been added successfully!”



Εικόνα Β.17: API HTTP POST postErrorFloater.php (πρόγραμμα Postman)

2.postErrorWater.php

```php
<?php
//POST error to table
    include_once('db_conn.php');
    if($_SERVER['REQUEST_METHOD'] == "POST"){
        // Get data from the REST client
        $task = isset($_POST['task']) ? mysqli_real_escape_string($conn, $_POST['task']) : "";
        $date = isset($_POST['date']) ? mysqli_real_escape_string($conn, $_POST['date']) : "";
        $priority = isset($_POST['priority']) ? mysqli_real_escape_string($conn, $_POST['priority']) : "";

        $id;
        $device_id;
        $device_code = 60;
        $description = 'Water Level is at maximum';


        //Get values from another table
        $sql1 = "SELECT * FROM device_status ORDER BY id DESC LIMIT 1";
        $get_data_query = mysqli_query($conn, $sql1) or die(mysqli_error($conn));
            if(mysqli_num_rows($get_data_query)!=0){

            if (mysqli_num_rows($get_data_query) > 0) {
              while($row = mysqli_fetch_assoc($get_data_query)) {
               $id = $row["id"];
               $device_id =  $row["device_id"];
              }
            }
        }

```

```php
        // Insert data into database
        $sql = "INSERT INTO `device_error`(`status_id`, `device_id`, `device_code`, `description`, `date`)
        VALUES ('.$id.','$device_id',60,'Water Level is at maximum',now())";


        $post_data_query = mysqli_query($conn, $sql);
        if($post_data_query){
            $json = array("status" => 1, "Success" => "To-Do has been added successfully!");
        }
        else{
            $json = array("status" => 0, "Error" => "Error adding To-Do! Please try again!");
        }
    }
    else{
        $json = array("status" => 0, "Info" => "Request method not accepted!");
    }
    @mysqli_close($conn);
    // Set Content-type to JSON
    header('Content-type: application/json');
    echo json_encode($json);
?>
```

Εικόνα Β.18: Κώδικας PHP postErrorWater.php

API: POST https://freatiot.000webhostapp.com/APIs/postErrorWater.php

Το API αυτό χρησιμοποιείται για να γραφτεί στην βάση, στον πίνακα device_error, το error στον αισθητήρα νερού. Μέσω Postman, το response του POST API είναι " To-Do has been added successfully!"

3. RainingCurrently.php

```php
<?php
    include_once('db_conn.php');
    //prepare sql statement

  if($_SERVER['REQUEST_METHOD'] == "POST")
    {
    // Get data from the REST client
        $task = isset($_POST['task']) ? mysqli_real_escape_string($conn, $_POST['task']) : "";
        $date = isset($_POST['date']) ? mysqli_real_escape_string($conn, $_POST['date']) : "";
        $priority = isset($_POST['priority']) ? mysqli_real_escape_string($conn, $_POST['priority']) : "";

        $RainingNow = $_GET['RainingNow'];
        $CurrentDate = $_GET["CurrentDate"];
        $isFloated = $_GET["isFloated"];


     $sql = "UPDATE `device_status` SET `dateAfter30`='".$CurrentDate."', `RainingBool`='".$RainingNow."',
     `dateAfterWater`='".$dateAfterWater."', `isFloated`='".$isFloated."' ORDER BY id DESC LIMIT 1";

        $post_data_query = mysqli_query($conn, $sql);
        if($post_data_query){
            $json = array("status" => 1, "Success" => "To-Do has been added successfully!");
        }
        else{
            $json = array("status" => 0, "Error" => "Error adding To-Do! Please try again!");
        }
    }

    else{
        $json = array("status" => 0, "Info" => "Request method not accepted!");
    }

    @mysqli_close($conn);
    // Set Content-type to JSON
    header('Content-type: application/json');
    echo json_encode($json);
?>
```

Εικόνα Β.19: Κώδικας PHP RainingCurrently.php

API: POST
https://freatiot.000webhostapp.com/APIs/RainingCurrently.php?RainingNow=0&CurrentDate=0&Floated=0

Το συγκεκριμένο API χρησιμοποιείται για να κάνει update τα παρακάτω πεδία της βάσης δεδομένων. Δηλαδή σε περίπτωση βροχής, το RainingBool γίνεται update μέσω της αυτού του API

+ Επιλογές

	id	device_id	humidity	waterSensor	FloaterSensor	isFloated	date	dateAfter30	RainingBool	UpdateButton
Επεξεργασία Αντιγραφή Διαγραφή	70	imei24	80	450	1	1	2020-12-08 23:18:11	0000-00-00 00:00:00	0	0
Επεξεργασία Αντιγραφή Διαγραφή	71	imei24	80	450	1	1	2020-12-08 23:18:11	0000-00-00 00:00:00	0	0
Επεξεργασία Αντιγραφή Διαγραφή	72	imei24	80	450	1	1	2020-12-08 23:18:11	0000-00-00 00:00:00	0	0
Επεξεργασία Αντιγραφή Διαγραφή	73	imei24	80	450	1	1	2020-12-08 23:18:11	0000-00-00 00:00:00	0	0
Επεξεργασία Αντιγραφή Διαγραφή	74	imei24	80	450	1	1	2020-12-08 23:18:11	0000-00-00 00:00:00	0	0
Επεξεργασία Αντιγραφή Διαγραφή	75	imei24	80	450	1	1	2020-12-08 23:18:11	0000-00-00 00:00:00	0	0
Επεξεργασία Αντιγραφή Διαγραφή	76	imei24	80	450	0	0	2020-12-08 23:18:11	2020-12-13 01:14:51	0	0
Επεξεργασία Αντιγραφή Διαγραφή	77	imei24	200	500	127	0	0000-00-00 00:00:00	0000-00-00 00:00:00	0	0
Επεξεργασία Αντιγραφή Διαγραφή	78	imei24	80	450	0	1	2020-12-08 23:18:11	0000-00-00 00:00:00	0	0
Επεξεργασία Αντιγραφή Διαγραφή	79	imei24	80	450	1	1	2020-12-08 23:18:11	0000-00-00 00:00:00	0	0
Επεξεργασία Αντιγραφή Διαγραφή	80	imei24	80	410	1	0	2021-01-06 09:18:11	2021-01-06 18:12:03	0	0

Εικόνα Β.20 : Πίνακας device_status

4. Write_data.php

```php
<?php
    include_once('db_conn.php');

    if (isset($_GET['sensor1']) || isset($_GET['sensor2']) )
    {
    // Get data from the REST client
        $task = isset($_POST['task']) ? mysqli_real_escape_string($conn, $_POST['task']) : "";
        $date = isset($_POST['date']) ? mysqli_real_escape_string($conn, $_POST['date']) : "";
        $priority = isset($_POST['priority']) ? mysqli_real_escape_string($conn, $_POST['priority']) : "";

        $sensorData1 = $_GET['sensor2'];
        $sensorData2 = $_GET['sensor3'];
        $sensorData3 = $_GET['sensor4'];
        //$sensorData2 = $_GET['sensor3'];
        $dateCreated = $_GET['created'];

                //Get values from another table
        $sql1 = "SELECT * FROM device_status ORDER BY id DESC LIMIT 1";
        $get_data_query = mysqli_query($conn, $sql1) or die(mysqli_error($conn));
            if(mysqli_num_rows($get_data_query)!=0){

            if (mysqli_num_rows($get_data_query) > 0) {
              while($row = mysqli_fetch_assoc($get_data_query)) {
                $isFloated = $row["isFloated"];
                $dateAfter30 = $row["dateAfter30"];
                $dateAfterWater =  $row["dateAfterWater"];
                $RainingBool =  $row["RainingBool"];
                $weatherStatus =  $row["weatherStatus"];
                $UpdateWater =  $row["UpdateWater"];
              }
            }
        }

        if ($sensorData3 == 0)
        {
            $sensorData4 = 0;
        }
        else
        {
            $sensorData4 = $isFloated;
        }
```

```php

        //na pernei tin proigoumeni timi twn dateAfter30 and rainingbool
        $sql = "INSERT INTO `device_status` (`device_id`, `humidity`, `waterSensor`, `FloaterSensor`, `isFloated`, `date`, `
            dateAfter30`, `dateAfterWater`, `RainingBool`, `weatherStatus`, `UpdateWater`)
        VALUES ('imei24', '".$sensorData1."', '".$sensorData2."', '".$sensorData3."', '".$sensorData4."', '".$dateCreated."', '
            ".$dateAfter30."', '".$dateAfterWater."', '".$RainingBool."', '".$weatherStatus."', '".$UpdateWater."')";

        $post_data_query = mysqli_query($conn, $sql);
        if($post_data_query){
            $json = array("status" => 1, "Success" => "To-Do has been added successfully!");
        }
        else{
            $json = array("status" => 0, "Error" => "Error adding To-Do! Please try again!");
        }
    }

    else{
        $json = array("status" => 0, "Info" => "Request method not accepted!");
    }

    @mysqli_close($conn);
    // Set Content-type to JSON
    header('Content-type: application/json');
    echo json_encode($json);

    //$conn->close();
?>
```

Εικόνα Β.21: Κώδικας PHP Write_data.php

API: POST
https://freatiot.000webhostapp.com/APIs/write_data.php?sensor2=4&sensor4=0&dateCreated=2021-01-06 18:12:01

Αφού γίνει η μέτρηση από το Arduino, καλείται αυτό το API ώστε να γραφτούν στην βάση οι μετρήσεις, στον πίνακα device_status.

Όμως, πριν προσθέσουμε τις νέες μετρήσεις, καλούμε ένα sql query όπου αποθηκεύουμε τις τελευταίες τιμές των dateAfter30, dateAfterWater. RainingBool, weatherStatus και updateWater και αυτές τις τιμές γράφονται στην βάση μαζί με τις μετρήσεις των αισθητηρίων. Αυτό γίνεται γιατί αν έρθει μια καινουργια μέτρηση, θα χαθούν τα δεδομένα αυτών των στηλών.

5. postUserMsg.php

```php
<?php
//POST error to table
    include_once('db_conn.php');
    if($_SERVER['REQUEST_METHOD'] == "POST"){
        // Get data from the REST client
        $task = isset($_POST['task']) ? mysqli_real_escape_string($conn, $_POST['task']) : "";
        $date = isset($_POST['date']) ? mysqli_real_escape_string($conn, $_POST['date']) : "";
        $priority = isset($_POST['priority']) ? mysqli_real_escape_string($conn, $_POST['priority']) : "";

        $Msg = $_GET['msg'];
        $id;
        $device_id;


        //Get values from another table
        $sql1 = "SELECT * FROM device_status ORDER BY id DESC LIMIT 1";
        $get_data_query = mysqli_query($conn, $sql1) or die(mysqli_error($conn));
            if(mysqli_num_rows($get_data_query)!=0){

            if (mysqli_num_rows($get_data_query) > 0) {
              while($row = mysqli_fetch_assoc($get_data_query)) {
               $id = $row["id"];
               $device_id =  $row["device_id"];
              }
            }
        }

        // Insert data into database
        $sql = "INSERT INTO `user_messages`(`status_id`, `device_id`, `message`, `date`)
        VALUES ('.$id.','$device_id','$Msg',now())";


        $post_data_query = mysqli_query($conn, $sql);
        if($post_data_query){
            $json = array("status" => 1, "Success" => "To-Do has been added successfully!");
        }
        else{
            $json = array("status" => 0, "Error" => "Error adding To-Do! Please try again!");
        }
    }
    else{
        $json = array("status" => 0, "Info" => "Request method not accepted!");
    }
    @mysqli_close($conn);
    // Set Content-type to JSON
    header('Content-type: application/json');
    echo json_encode($json);

?>
```

Εικόνα Β.22: Κώδικας postUserMsg.php

API: POST https://freatiot.000webhostapp.com/APIs/postUserMsg.php?msg=Level of water is high, but floater is 0. Please Check.

Με αυτό το API, γράφονται στην βάση τα user messages, δηλαδή τα μηνύματα που έχουν να κάνουν με τις δυσλειτουργίες της συσκευής.

ΠΑΡΑΡΤΗΜΑ Γ: ΛΕΙΤΟΥΡΓΊΑ ΚΩΔΙΚΑ ΣΕΛΙΔΑΣ

Ο κώδικας κι η λογική με την οποία δημιουργήθηκαν τα API μπορεί να διαπιστωθεί στο παράρτημα Β. Σε αυτό το παράρτημα, όμως, θα μιλήσουμε για το JavaScript και AJAX κομμάτι όπου σε αυτό το σημείο γίνονται τα HTTP Requests και καλούνται τα APIs.

Αρχικά πρώτα θα δούμε το script στο οποίο παίρνουμε τις μετρήσεις των αισθητηρίων από το cloud και στην συνέχεια πως τα αποθηκεύουμε στην βαση.

```
<script>
        var last_date = '';

        $(document).ready( function() {
            pingInsertStatus();
        });

        function pingInsertStatus() {
            $.ajax({
                type: "GET",
                url: "https://api.thingspeak.com/channels/1279772/feeds.json?results=1",
                cache: false,
                async: true,
                success: function(data){
                    res6 = data.feeds[0];
                    sensor2 = res6.field2;
                    sensor3 = res6.field3;
                    sensor4 = res6.field4;
                    date= res6.created_at;
                    date = date.replace("T", " ");
                    date = date.replace("Z", "");
                    date1 = new Date (date);

                    $.ajax({
                        type: "GET",
                        url: "http://localhost/ptixiaki%20v1/APIs/info.php",
                        cache: false,
                        async: true,
                        success: function (data) {
                            res = data.info[0];
                            last_date= res.date
                            last_date1 = new Date(last_date);


                            if (date1.getTime() !== last_date1.getTime()) { //!==
                                $.ajax({
                                    type: "POST",
                                    url: "http://localhost/ptixiaki%20v1/APIs/write_data.php?sensor2="
                                    +sensor2+"&sensor3="+sensor3+"&sensor4="+sensor4+"&created="+date,
                                    success: function (data) {
                                        console.log("Data of sensors are inserted in DB");
                                    }
                                });

                            }
                        }
                    });
                     setTimeout("pingInsertStatus()", 30000);
                    }
```

Εικόνα Γ.1: Κώδικας AJAX αποθήκευσης δεδομένων από το Cloud στην βάση

Αρχικά χρησιμοποιούμε το AJAX με type GET, ώστε να κάνουμε ένα get http request από το thingspeak, που είναι το cloud service. Αποθηκεύουμε τις μετρήσεις που παίρνουμε από το API

στις μεταβλητές sensor2 όπου αποθηκεύεται η μέτρηση της υγρασίας από το DHT11, sensor3 όπου αποθηκεύεται η μέτρηση του αισθητήρα στάθμης νερού, sensor4 όπου αποθηκεύεται η τιμή του φλοτέρ και date όπου αποθηκεύεται η ημερομηνία της μέτρησης.

Στην συνέχεια καλούμε το API info.php όπου όπως είπαμε στο Παράρτημα Β, επιστρέφει τις τελευταίες μετρήσεις της συσκευής. Από αυτό το API παίρνουμε την ημερομηνία. Οπότε συγκρινονται η ημερομηνία που υπάρχει στο cloud και η ημερομηνία της τελευταίας μέτρησης. Αν οι δύο τιμές είναι διαφορετικές, τότε καλώντας το POST API write_data.php, οι νέες μετρήσεις εισάγονται στην βάση.

Αυτή η συνάρτηση τρέχει καθε 30000ms, δηλαδή κάθε 30 δευτερόλεπτα.

Στο επόμενο script, καλείται το το API info.php όπου συλλέγονται οι τελευταίες μετρήσεις της συσκευής και εμφανίζονται στην ιστοσελίδα, στον πίνακα "Πληροφορίες Φρεατίου"

```
237     <script>
238         //console.log("Display Device Status");
239
240
241         $(document).ready( function() {
242             pingDisplayStatus();
243         });
244
245
246         var lastNotificationIdentifier01 = '2000-01-01 00:00:00';
247         lastNotificationIdentifier0 = new Date(lastNotificationIdentifier01);
248         //GET freation info from DB
249         function pingDisplayStatus() {
250             $.ajax({
251                 type: "GET",
252                 url: "http://localhost/ptixiaki%20v1/APIs/info.php",
253                 cache: false,
254                 async: true,
255                 success: function(data){
256                     res = data.info[0];
257                     localIdentifier01 = res.date;
258                     localIdentifier0 = new Date(localIdentifier01);
259                     isFloatedSensor = res.FloaterSensor;
260                     waterLevel = res.waterSensor;
261                     isFloated = res.isFloated;
262
263
264                     // When a new row is added, then table is going to be updated with newest data
265                     if (localIdentifier0.getTime() !== lastNotificationIdentifier0.getTime() ) {
266                         //Display info in Details table
267                         document.getElementById("Id").innerHTML = res.id;
268                         document.getElementById("DevId").innerHTML= res.device_id;
269                         document.getElementById("UltraSen").innerHTML= res.humidity;
270                         document.getElementById("WatSen").innerHTML= res.waterSensor;
271                         document.getElementById("Flooded").innerHTML= res.FloaterSensor;
272                         document.getElementById("date").innerHTML= res.date;
273
274                         lastNotificationIdentifier0 = localIdentifier0;
275
276                     }
277
```

Εικόνα Γ.2: Κώδικας AJAX εμφάνισης δεδομένων στον πίνακα "Πληροφορίες φρεατίου"

Για να επιτευχθεί αυτό συγκρίνουμε την ημερομηνία της τελευταίας μέτρησης με την μεταβλητή lastNotificationIdentifier0. Αν είναι διαφορετικές οι τιμές, τότε στην ιστοσελίδα εμφανίζονται οι νέες μετρήσεις.

Στο ίδιο script βρίσκεται και ο παρακάτω κώδικας, όπου έχει να κάνει με τα error messages που εμφανίζονται πάνω από τον χάρτη σε περίπτωση ενεργοποίησης των αισθητηρίων.

```
278                         //According to isFloated, display error message
279                         if (isFloatedSensor == 1) {
280                            document.getElementById("userMsg").style.display = "block";
281                            document.getElementById("img_flo").style.display = "block";
282                            document.getElementById('userMsg').style.backgroundColor = '#FFA500';
283 
284                            //currentTime = new Date();
285                            //console.log(currentTime);
286                         }
287                         else if (isFloatedSensor == 0)
288                         {
289                            document.getElementById("userMsg").style.display = "none";
290                            document.getElementById("img_flo").style.display = "none";
291                            document.getElementById('userMsg').style.backgroundColor = '#FFA500';
292                         }
293 
294 
295                         if (waterLevel >= 500)
296                         {
297                             document.getElementById("waterlevel").style.display = "block";
298                             document.getElementById("img_wat").style.display = "block";
299                             document.getElementById('userMsg').style.backgroundColor = '#FFA500';
300                         }
301                         else
302                         {
303                            document.getElementById("waterlevel").style.display = "none";
304                            document.getElementById("img_wat").style.display = "none";
305                            document.getElementById('userMsg').style.backgroundColor = '#FFA500';
306                         }
307 
308                         if (isFloated == 1)
309                         {
310                            document.getElementById("timer1").innerHTML = "Timer τελείωσε.Το  Φρεάτιο είναι βουλωμένο";
311                            document.getElementById("timer").innerHTML = "";
312                            document.getElementById('userMsg').style.backgroundColor = 'red';
313                         }
314 
315 
316 
317                     setTimeout("pingDisplayStatus()", 5000);
318                 }
319             });
320         };
321     </script>
```

Εικόνα Γ.3: Κώδικας AJAX εμφάνισης μηνυμάτων στην σελίδα

Δηλαδή, αν το φλοτέρ έχει την τιμή 1 τότε εμφανίζεται το μήνυμα που αναφέρει ότι "το φρεάτιο εμφανίζεται βουλωμένο". Αντιθέτως, αν η τιμή είναι 0 τότε δεν εμφανίζεται τίποτα. Αντίστοιχα και για τον αισθητήρα στάθμης νερού ακολουθείτε η ίδια λογική, δηλαδή αν είναι πάνω ή ίσο απο 500 εμφανίζεται το μήνυμα ενω αν είναι κάτω από 500 δεν εμφανίζεται μήνυμα. Η τελευταία συνθήκη (isFloated ==1), γίνεται true μετά το timer των 30 λεπτών, όπου εμφανίζεται το μήνυμα που λέει "Timer τελείωσε. Το φρεάτιο ειναι βουλωμένο." και το χρώμα μηνύματος γίνεται κόκκινο.

Το script αυτό τρέχει κάθε 5000ms, δηλαδή 5 δευτερόλεπτα.

Το επόμενο script έχει να κάνει με τα δεδομένα που εμφανίζονται στην ιστοσελίδα, στον πίνακα Σφάλματα και User messages συλλέγοντας τα δεδομένα από το API info_error.php και info_messages.php, αντίστοιχα.

```
326         $(document).ready( function() {
327             pingErrorLogs();
328             pingUserMsgLogs();
329         });
330       function pingErrorLogs() {
331                 var table = $('#table_error').DataTable( {
332                 ajax: {
333                    url: 'http://localhost/ptixiaki%20v1/APIs/info_error.php',
334                    method: 'GET',
335                    xhrFields: {
336                        withCredentials: true
337                    },
338                    dataSrc:'info',
339                 },
340                 "order": [[ 0, "desc" ]],
341                 columns: [
342                     { data: 'date' },
343                     { data: 'device_id' },
344                     { data: 'description'}
345                 ]
346                 } );
347 
348                 setInterval( function () {
349                 table.ajax.reload();
350                 }, 30000 );
351 
352          }
353       function pingUserMsgLogs() {
354             var table = $('#table_msg').DataTable( {
355             ajax: {
356                url: 'http://localhost/ptixiaki%20v1/APIs/info_messages.php',
357                method: 'GET',
358                xhrFields: {
359                    withCredentials: true
360                },
361                dataSrc:'info',
362             },
363             "order": [[ 0, "desc" ]],
364             columns: [
365                 { data: 'date' },
366                 { data: 'device_id' },
367                 { data: 'message'}
368             ]
369             } );
370 
371             setInterval( function () {
372             table.ajax.reload();
373             }, 30000 );
```

Εικόνα Γ.4: Κώδικας AJAX εμφάνισης μηνυμάτων στο πίνακα Σφάλματα και User Messages

Αυτές οι δύο συναρτήσεις καλούνται κάθε 30000ms, δηλαδή κάθε 30 δευτερόλεπτα.

Το επόμενο script αρχικά καλεί το API freatioStatus.php για να πάρει τις τελευταίες τιμές του αισθητήρα φλοτέρ και του flag “RainingNow”.

```
391     function pingStatus30min() {
392      $.ajax({
393          type: "GET",
394          url: "http://localhost/ptixiaki%20v1/APIs/freatioStatus.php",
395          cache: false,
396          async: true,
397          success: function(data){
398              res1 = data.info[0];
399              localIdentifier1 = res1.id;
400              isFloated30 = res1.FloaterSensor;
401              Rainingbool = res1.RainingNow;
402              //console.log(isFloated30);
403                  $.ajax({
404                          type: "GET",
405                          //url: "http://localhost/ptixiaki%20v1/APIs/info.php",
406                          url: "http://api.openweathermap.org/data/2.5/weather?lat=38.05&lon=23.8&units=metric&appid=8f6651468a6cbf13dff0eaaa5f7d6e9c",
407                          cache: false,
408                          async: true,
409                          success: function (data) {
410                              res = data.weather[0]; //res = data.info[0];
411                               desc = res.main;   // //Raining = res.RainingBool;
412                              console.log(desc);
413                              if (desc == "Thunderstorm" || desc == "Rain" || desc == "Drizzle")
414                              {
415
416                              if (isFloated30 == 1) {
417
418                                  document.getElementById("add_freatio").innerHTML ="Βρέχει αυτή την στιγμή.";
419                                  document.getElementById("add_water").innerHTML ="";
420
421                                  console.log ("Its raining currently");
422                              }
423
424                                  console.log("Its raining currently");
425                                   $.ajax({
426                                      type: "POST",
427                                      url: "http://localhost/ptixiaki%20v1/APIs/RainingCurrently.php?RainingNow=1&CurrentDate=0",
428                                      success:function (data) {
429                                          console.log("Its raining currently");
430                                      }
431                                  });
432                              }
433                              else
434                              {
435                                 if (isFloated30 == 1) {
436
437                                  document.getElementById("add_freatio").innerHTML ="Όμως δεν βρέχει";
438                                  document.getElementById("add_water").innerHTML ="όμως δεν βρέχει.";
439                              }
```

Εικόνα Γ.5: Κώδικας AJAX για τις καιρικές συνθήκες

Στην συνέχεια καλείται το API του καιρού, όπου αν βρέχει και το φλοτέρ έχει την τιμή 1, τότε προστίθεται στο μήνυμα (πάνω από τον χάρτη) ότι “Βρέχει αυτή την στιγμή”. Αν δεν βρέχει, τότε προστίθεται το μήνυμα “Όμως δεν βρέχει”. Επίσης ενώ βρέχει καλείται το post API που κάνει update στην βάση το RainingFlag σε 1.

Στην επόμενη εικόνα, εφόσον το RainingFlag ειναι 1, έχει σταματήσει να βρέχει και η τιμή του φλοτέρ αισθητήρα είναι 0 (isFloated30 ==0) τότε καλείται το API NoMeasure.php όπου τσεκάρει τις τελευταίες 200 μετρήσεις. Αν το άθροισμα των μετρήσεων του αισθητήρα στάθμης νερού και του αισθητήρα φλοτέρ είναι 0, τότε η τιμή του error είναι 1. Συνεπώς, καλείται το POST API PostUserMsg.php όπου εισάγετε στην βάση (στον πίνακα user_messages) μια νέα γραμμή που θα έχει σαν μήνυμα “No data was registered during rain. Please Check.”

```
444
445                 if (Rainingbool == 1 && ( desc == "Snow" || desc == "Clouds" || desc == "Clear" || desc == "Atmosphere"))
446                 {
447                  if (isFloated30 == 0)
448                  {
449                     $.ajax({
450                         type: "GET",
451                         url: "http://localhost/ptixiaki%20v1/APIs/NoMeasure.php",
452                         cache: false,
453                         async: true,
454                         success: function(data){
455                             res1 = data.info[0];
456                             error = res1.error;
457                             if (error == 1)
458                             {
459                                 $.ajax({
460                                     type: "POST",
461                                     url: "http://localhost/ptixiaki%20v1/APIs/PostUserMsg.php?msg=No data was registered during rain. Please Check.",
462                                     success: function (data) {
463                                         console.log("UserMsg is Posted.");
464                                          }
465                                     });
466                             }
467                          }
468                     });
469                  }
```

Εικόνα Γ.6: Κώδικας AJAX για εισαγωγη δεδομένων στο User Messages

Στην επόμενη εικόνα, εφόσον το RainingFlag ειναι 1, σταμάτησε να βρέχει και το η τιμή του φλοτέρ αισθητήρα είναι ακόμα 1 (isFloated30 ==1) τότε αποθηκεύεται η τωρινή ώρα κι πάνω σε αυτήν προστίθενται 30 λεπτά και αποθηκεύεται στην βάση δεδομένων χρησιμοποιώντας το POST API RainingCurrently.php. Παράλληλα στην βάδη δεδομένων το RainingNow flag παίρνει την τιμή 0.

```
503                     if (isFloated30 == 1 && isFloated == 0)
504                     {
505                        console.log ("Its NOT raining anymore");
506                        const format1 = "YYYY-MM-DD HH:mm:ss"
507                        currentTime = new Date();
508                        console.log(currentTime);
509                        var currentTimeN30 = moment(currentTime).add(1, 'm').toDate(); //add 30
510                        console.log(currentTimeN30);
511                        currentTimeN30 = moment(currentTimeN30).format(format1);
512                        console.log(currentTimeN30);
513
514                        $.ajax({
515                                 type: "POST",
516                                 url: "http://localhost/ptixiaki%20v1/APIs/
                                     RainingCurrently.php?RainingNow=2&CurrentDate="+currentTimeN30+"&isFloated=0
                                     ",
517                                 success: function (data) {
518                                     console.log("Rain is over. Register Date.");
519                                 }
520                        });
521
522                       }
523
524                     }
525
526                 }
527              });
528
529
530           setTimeout("pingStatus30min()", 10000);
531         }
```

Εικόνα Γ.7: Κώδικας AJAX για εισαγωγή της προστιθέμενης ώρας στην βάση

Αυτή η συνάρτηση καλείται κάθε 10000ms, δηλαδή 10 δευτερόλεπτα.

Στο επόμενο script, πραγματοποιείται το countdown για να αποφανθεί αν όντως το φρεάτιο είναι φραγμένο.

Πρώτα καλείται το GET API freatioStatus.php για να πάρουμε τα τελευταία δεδομένα του αισθητήρα επίπλευσης, την ημερομηνία με τα προστιθέμενα 30 λεπτά και το Rain Flag.
Αν η if συνθήκη είναι true, τότε ξεκινάει το countdown κι εμφανίζεται το μήνυμα "Παρακαλώ ελέγξτε μετά το countdown: m s ". Η συνάρτηση του countdown καλείται κάθε 1 δευτερόλεπτο. Αφού τελειώσει το countdown η τιμή endTimer γίνεται 1.

```
505     <script>
506         var endTimer = 0;
507         $(document).ready( function() {
508             FreatioIsFloaded();
509         });
510 
511         // Freatio is floaded
512        function FreatioIsFloaded() {
513         $.ajax({
514             type: "GET",
515             url: "http://localhost/ptixiaki%20v1/APIs/freatioStatus.php",
516             cache: false,
517             async: true,
518             success: function(data){
519                 res1 = data.info[0];
520                 localIdentifier1 = res1.id;
521                 floaterSensor = res1.FloaterSensor
522                 EndOfRain = res1.dateAfter30;
523                 RainFlag = res1.RainingBool;
524                 //console.log(RainFlag);
525                 countDownDate = new Date(EndOfRain).getTime();
526                 console.log(countDownDate);
527                 isFloated = res1.isFloated
528               if (floaterSensor == 1 && countDownDate > 0 && isFloated == 0 && RainFlag == 1)
529               {
530                 var x = setInterval(function() {
531                       // Get today's date and time
532                       var now = new Date().getTime();
533 
534                       // Find the distance between now and the count down date
535                       var distance = countDownDate - now;
536 
537                       // Time calculations for days, hours, minutes and seconds
538                       var minutes = Math.floor((distance % (1000 * 60 * 60)) / (1000 * 60));
539                       var seconds = Math.floor((distance % (1000 * 60)) / 1000);
540 
541                       // Output the result in an element with id="demo"
542                       document.getElementById("timer").innerHTML ="Παρακαλώ ελέγξτε μετά το countdown:" + minutes + "m " + seconds + "s ";
543 
544                       // If the count down is over, call FreatioStatus. If its still floated
545                       if (distance < 0) {
546                         clearInterval(x);
547                         document.getElementById("timer").innerHTML = "";
548                         console.log("enter1");
549                         endTimer = 1;
550                         //document.getElementById('userMsg').style.backgroundColor = 'red';
551 
552                       }
553                 }, 1000);
```

Εικόνα Γ.8: Κώδικας AJAX για το timer του αισθητήρα φλοτέρ

Έπειτα εφόσον η συνθήκη endTimer == 1 είναι true, καλούνται δύο POST APIs, το ένα για να εισαχθεί στη βάση το σφάλμα "Floated is Blocked" στον πίνακα device_error και το άλλο κάνει update το value της βάσης isFloated σε 1, που σημαίνει ότι όντως το φρεάτιο είναι φραγμένο.

```
555                 if (endTimer == 1) {
556                     console.log("enter12");
557                         $.ajax({
558                                 type: "POST",
559                                 url: "http://localhost/ptixiaki%20v1/APIs/RainingCurrently.php?RainingNow=0&CurrentDate="+EndOfRain+"&isFloated=1",
560                                 success: function (data) {
561                                     console.log("Rain is over. Register Date.");
562                                   }
563                         });
564
565
566                         $.ajax({
567                             type: "POST",
568                             url: "http://localhost/ptixiaki%20v1/APIs/postErrorFloater.php",
569                             success: function (data) {
570                             console.log("Data in ERROR table is posted");
571                             endTimer = 0;
572                             }
573                         });
574
575
576                 }
577
578               }
579                  setTimeout("FreatioIsFloaded()", 10000);
580                 }
581
582             });
583         }
```

Εικόνα Γ.9: Κώδικας AJAX για την προσθήκη σφάλματος για τον αισθητήρα φλοτέρ

Η συνάρτηση αυτή καλείται κάθε 10000 ms, δηλαδή κάθε 10 δευτερόλεπτα.

Το επόμενο script έχει να κάνει με τον αισθητήρα στάθμης νερού. Πρώτα καλείται το API του freatioStatus.php όπου παίρνουμε τις τιμές του αισθητήρα στάθμης νερού, το timer του στάθμης νερού και την ημερομηνία της τελευταία μέτρησης. Στην συνέχεια αν η τιμή του waterLevel είναι κάτω από 500 τότε ο αισθητήρας είναι σε καλή κατάσταση και καλείται το POST API όπου μηδενίζει το timer και την μεταβλητή update.
Αν η τιμή του είναι πάνω από 500 τότε παίρνουμε την τωρινή ώρα και προστίθενται σε αυτήν 20 λεπτά (στα πλαίσια της διπλωματικής βάλαμε 1 λεπτό) και με POST API αποθηκεύουμε την ώρα στην βάση κι αλλάζει η μεταβλητή του update σε 1.

```
598     function PostErrorWater() {
599      $.ajax({
600          type: "GET",
601          url: "http://localhost/ptixiaki%20v1/APIs/freatioStatus.php",
602          cache: false,
603          async: true,
604          success: function(data){
605              res1 = data.info[0];
606              waterLevel = res1.waterSensor;
607              afterTimer = res1.dateAfterWater;
608              date  = res1.date
609              UpdateWater = res1.UpdateWater;
610              FloaterSensor =res1.FloaterSensor;
611              console.log(afterTimer);
612              if (waterLevel <= 500)
613              {
614                  post = 0;
615                  console.log(post);
616                   if (UpdateWater == 1) {
617                     var  currentTimeN30 = "0000-00-00 00:00:00";
618                     $.ajax({
619                              type: "POST",
620                              url: "http://localhost/ptixiaki%20v1/APIs/addTimer.php?timer="+currentTimeN30+"&update=0",
621                              success: function (data) {
622                                  console.log("Water Sensor works properly");
623                                  post = 2;
624                              }
625                     });
626                    }
627              }
628              if (waterLevel > 500) {
629
630                   if (UpdateWater == 0 ) {
631                     const format1 = "YYYY-MM-DD HH:mm:ss"
632                     currentTime = new Date();
633                     var currentTimeN30 = moment(currentTime).add(1, 'm').toDate(); //add 30
634                     currentTimeN30 = moment(currentTimeN30).format(format1);
635                     console.log("For water level sensor:");
636                     console.log(currentTimeN30);
637
638                     $.ajax({
639                              type: "POST",
640                              url: "http://localhost/ptixiaki%20v1/APIs/addTimer.php?timer="+currentTimeN30+"&update=1",
641                              success: function (data) {
642                                  console.log("Water Sensor Timer added.");
643                                  post = 2;
644                              }
645                     });
646                   }
```

Εικόνα Γ.10: Κώδικας AJAX για την προσθήκη του timer για τον αισθητήρα στάθμης

Στην συνέχεια ξεκινά η συνάρτηση με το timer για τον αισθητήρα στάθμης νερού όπου καλείται κάθε 1 δευτερόλεπτο.

```
648                 countDownDate1 = new Date(afterTimer).getTime();
649                 if (countDownDate1 > 0 && waterLevel > 500 && UpdateWater == 1)
650                 {
651                  console.log("countdown");
652                       var x = setInterval(function() {
653                       // Get today's date and time
654                       var now1 = new Date().getTime();
655
656                       // Find the distance between now and the count down date
657                       var distance1 = countDownDate1 - now1;
658
659                       // Time calculations for days, hours, minutes and seconds
660                       var minutes1 = Math.floor((distance1 % (1000 * 60 * 60)) / (1000 * 60));
661                       var seconds1 = Math.floor((distance1 % (1000 * 60)) / 1000);
662
663                       // Output the result in an element with id="demo"
664                       document.getElementById("add_water1").innerHTML ="Παρακαλώ ελέγξτε μετά το countdown:" + minutes1 + "m " + seconds1 + "s ";
665
666                       // If the count down is over, call FreatioStatus. If its still floated
667                       if (distance1 < 0) {
668                         clearInterval(x);
669                         document.getElementById("add_water1").innerHTML = "Timer Done. Το φρεάτιο ειναι πλημμυρισμένο μέχρι την 1/2.";
670                         //document.getElementById('waterlevel').style.backgroundColor = 'red';
671                         endTimer1 = 1;
672                         //document.getElementById('userMsg').style.backgroundColor = 'red';
673
674                       }
675                  }, 1000);
```

Εικόνα Γ.11: Κώδικας AJAX για το timer για τον αισθητήρα στάθμης

Τέλος αφού τελειώσει το timer, πρώτα τσεκάρουμε να δούμε αν η τιμή του floater είναι 0. Αν είναι, τότε καλείται το API PostUserMsg.php όπου εισάγετε στην βάση (στον πίνακα user_messages) μια νέα γραμμή που θα έχει σαν μήνυμα “Level of water is high, but floater is 0. Please Check.”

Επιπροσθέτως, καλείται το POST API postErrorWater.php όπου στο πίνακα device_error εισάγεται το σφάλμα “Water Level is at maximum” και παράλληλα καλείται το POST API addTimer.php όπου μηδενίζει το timer του στάθμης νερού.

```
                    if (endTimer1 == 1 && (afterTimer != "2000-01-01 00:00:00")) {
                        if (FloaterSensor == 0)
                        {

                            $.ajax({
                            type: "POST",
                            url: "http://localhost/ptixiaki%20v1/APIs/PostUserMsg.php?msg=Level of water is high, but floater is 0. Please Check.",
                            success: function (data) {
                                console.log("UserMsg is Posted.");
                                 }
                            });
                        }

                        $.ajax({
                                type: "POST",
                                url: "http://localhost/ptixiaki%20v1/APIs/postErrorWater.php",
                                success: function (data) {
                                    console.log("Posted Error for Floater");
                                    endTimer1 = 0;
                                    post == 0;
                                    var currentTimeN30 = "2000-01-01 00:00:00";
                                    $.ajax({
                                        type: "POST",
                                        url: "http://localhost/ptixiaki%20v1/APIs/addTimer.php?timer="+currentTimeN30+"&update=1",
                                        success: function (data) {
                                            console.log("Change date to 0");
                                            post = 2;
                                        }
                                    });
                                     }
                        });
                    }

                }
            }
            console.log(post);
            setTimeout("PostErrorWater()", 5000);

        }
```

Εικόνα Γ.12: Κώδικας AJAX μετά το timer για τον αισθητήρα στάθμης

Τέλος για τον χάρτη Bing Maps, για να καταφέρουμε να αλλάξουμε το χρώμα του κύκλου στον χάρτη, καλείται το API freatioStatus.php και ανάλογα με την τιμή του floater αισθητήρα αλλάζει το χρώμα σε κόκκινο ή πράσινο.

```
$.ajax({
    contentType: "application/json",
    type: "GET",
    url: "http://localhost/ptixiaki%20v1/APIs/freatioStatus.php",
    success: function (data) {
          res = data.info[0];
          isFloated = res.FloaterSensor;
          //console.log(isFloated);
          if (isFloated == 1) {
            createColoredPushpin(map.getCenter(), 'red', function (pin) {
               map.entities.push(pin);
              //Store some metadata with the pushpin.
              pin.metadata = {
                title: 'Θηβών κι Ιερά Οδός 50',
                description: 'Status: Error '
              };

              //Add a click event handler to the pushpin.
              Microsoft.Maps.Events.addHandler(pin, 'click', pushpinClicked);

              });
            } else if (isFloated == 0)
            {
              createColoredPushpin(map.getCenter(), 'green', function (pin) {
                 map.entities.push(pin);
                //Store some metadata with the pushpin.
                pin.metadata = {
                  title: 'Θηβών κι Ιερά Οδός 50',
                  description: 'Status: OK'
                };

              //Add a click event handler to the pushpin.
              Microsoft.Maps.Events.addHandler(pin, 'click', pushpinClicked);
              });
            }

          //setTimeout("GetMap()", 5000)
        }
    });

}
```

Εικόνα Γ.13: Κώδικας AJAX για το κύκλο στο χάρτη

Να σημειωθεί ότι ο ολόκληρος ο κώδικας μπορεί να βρεθεί στην ιστοσελίδα: https://github.com/ellez1514/ptixiaki-v1

ΒΙΒΛΙΟΓΡΑΦΙΑ - ΠΗΓΕΣ

[1] Γενικη Γραμματεία Πολιτικής Προστασίας (2020), https://www.civilprotection.gr/el/imerisios-hartis-provlepsis-kindynoy-pyrkagias-31102020. (Ανακτήθηκε στις 10 Ιανουαρίου 2021)
[2] McLaren, Duncan; Agyeman, Julian (2015), Sharing Cities: A Case for Truly Smart and Sustainable Cities. MIT Press. ISBN 9780262029728.
[3] StartUpper.(2019), https://startupper.gr/vipnews/54629/auti-einai-i-startup-etaireia-pou-katharizei-tin-athina/ (Ανακτήθηκε στις 2 Φλεβάρη 2021)
[4] Δημοτική Επιχείρηση Ύδρευσης - Αποχύτευσης Εύβοιας, https://www.deyav.gr/i-devav/apoxeytefsi. (Ανακτήθηκε στις 3 Ιανουαρίου 2021)
[5] i-TECH4u, (2014) https://web.archive.org/web/20180202162428/http://www.itech4u.gr/tech/hands-on/item/7262-internet-of-things-se-apla-ellinika/7262-internet-of-things-se-apla-ellinika (Ανακτήθηκε στις 25 Ιανουαριου 2020)
[6] Mitchell, T. (1997). Machine Learning. McGraw Hill. p. 2. ISBN 978-0-07-042807-2.
[7] Arduino cc. https://www.arduino.cc/en/Guide/Introduction (Ανακτήθηκε στις 28 Δεκέμβρη 2020)
[8] Παπάζογλου, Λιώνης. (2018). Ανάπτυξη εφαρμογών με το Arduino 2η έκδοση. Εκδόσεις Τζιόλα. ISBN 9789604185504.
[9] developershome.com. https://www.developershome.com/sms/atCommandsIntro.asp (Ανακτήθηκε στις 28 Δεκέμβρη 2020)
[10] M2MSupport.net. https://m2msupport.net/m2msupport/atcipstart-start-up-tcp-or-udp-connection/ (Ανακτήθηκε στις 18 Δεκεμβρίου 2020)
[11] M2MSupport.net. https://m2msupport.net/m2msupport/atcipsend-send-data-through-tcp-or-udp-connection/ (Ανακτήθηκε στις 18 Δεκεμβρίου 2020)
[12] Postman, https://www.postman.com/company/about-postman/ (Ανακτήθηκε στις 20 Δεκεμβρίου 2020)
[13] Fritzing, https://fritzing.org/ (Ανακτήθηκε 5 Ιανουαρίου 2021)
[14] Datasheet ESP8266 Serial Esp-01 WIFI Wireless, https://components101.com/sites/default/files/component_datasheet/ESP8266%20Datasheet.pdf (Ανακτήθηκε 15 Ιανουαρίου 2021)
[15] Virtual Paradigm Online Free Edition, https://online.visual-paradigm.com/diagrams/solutions/free-visual-paradigm-online/ (Ανακτήθηκε 2 Φλεβάρη 2021)
[16]Thingspeak, https://thingspeak.com/(Ανακτήθηκε 2 Φλεβάρη 2021)

[17] JavatPoint.com,
https://www.javatpoint.com/php-session#:~:text=PHP%20session%20is%20used%20to,until%20user%20close%20the%20website).&text=PHP%20session%20creates%20unique%20user,avoid%20conflict%20between%20multiple%20browsers.
(Ανακτήθηκε στις 21 Φεβρουαρίου 2021)
[18] Bing Maps, https://www.microsoft.com/en-us/maps/developer-resources (Ανακτήθηκε στις 21 Φεβρουαρίου 2021)
[19] Open Weather Maps, https://openweathermap.org/api (Ανακτήθηκε στις 21 Φεβρουαρίου 2021)
[20] Εθνική Μετεωρολογική Υπηρεσία,
http://www.hnms.gr/emy/el/climatology/climatology_extreme (Ανακτήθηκε στις 18 Δεκεμβρίου 2020)
[21] Εθνικός Κήρυξ (2017),
https://www.ekirikas.com/archive_greece/arthro/plimmyres_stin_ellada_h_istoria_epanalambanetai-157635/ Ανακτήθηκε στις 18 Δεκεμβρίου 2020)
[22] Meteo.gr, https://www.meteo.gr/ClimaticData.cfm (Ανακτήθηκε στις 21 Φεβρουαρίου 2021)
[23] Εθνική Μετεωρολογική Υπηρεσία,
http://www.hnms.gr/emy/el/services/paroxi-ipiresion-klimatika-dedomena (Ανακτήθηκε στις 10 Γενάρη 2021)
[24] Meteo.gr, https://www.meteo.gr/ClimaticData.cfm (Ανακτήθηκε στις 10 Γενάρη 2021)
[25] Monkey Learn, https://monkeylearn.com/blog/classification-algorithms/. (Ανακτήθηκε στις 20 Ιανουαρίου 2021)
[26] Matlab, https://www.mathworks.com/products/matlab-online.html (Ανακτήθηκε στις 20 Ιανουαρίου 2021)