Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Μηχανικών Πληροφορικής

Πτυχιακή εργασία

Έξυπνο σύστημα αυτόματου ποτίσματος με απομακρυσμένο έλεγχο

Νικολουδάκης Γιάννης (ΑΜ: 766)

Επιβλέπων καθηγητής : Δρ. Παναγιωτάκης Σπυρίδων

ABSTRACT

The purpose of this project is to develop an intelligent watering system with a specific timetable, which will decide dynamically based on temperature and humidity, whether it should stop watering, or start, regardless of the schedule. The system will send data, such as temperature-humidity, soil moisture and system status on a remote server (mySQL) using GSM module (3G-2G). The entire system will be supervised and controlled remotely through an ANDROID application. The application will "read" the data from the SQL server and will interact with the system via SMS. Therefore, the user will be able to monitor the status of the field and dynamically start or stop watering as needed. The implementation of this system will be using the development board ARDUINO mega2560 and regional materials of the same "family".

ΣΥΝΟΨΗ

Ο σκοπός αυτής της εργασίας είναι η ανάπτυξη ενός έξυπνου συστήματος που θα "ποτίζει" με συγκεκριμένο χρονοδιάγραμμα, αλλά θα αποφασίζει δυναμικά βάση θερμοκρασίας και υγρασίας, αν θα πρέπει να διακόψει το πότισμα, ή να το ξεκινήσει ασχέτως του χρονοπρογράμματός του. Το σύστημα θα στέλνει δεδομένα, όπως, θερμοκρασία-υγρασία περιβάλλοντος, υγρασία χώματος και κατάσταση συστήματος σε έναν απομακρυσμένο server (mySQL) μέσω GSM module (3G-2G). Όλο το σύστημα, θα επιβλέπεται και θα ελέγχεται απομακρυσμένα, μέσω μιας εφαρμογής ANDROID. Η εφαρμογή, θα "διαβάζει" τα δεδομένα του SQL server και θα αλληλεπιδρά με το σύστημα μέσω SMS. Θα μπορεί λοιπόν ο χρήστης να επιβλέπει την κατάσταση του χωραφιού και θα μπορεί δυναμικά να ξεκινά ή να σταματά το πότισμα κατά το δοκούν. Η υλοποίηση αυτού του συστήματος θα γίνει με τη χρήση της αναπτυξιακής πλακέτας ARDUINO mega2560 και περιφερειακών υλικών της ίδιας οικογένειας.

ΚΙΝΗΤΡΟ ΓΙΑ ΤΗΝ ΔΙΕΞΑΓΩΓΗ ΤΗΣ ΕΡΓΑΣΙΑΣ

Είναι καθολικά γνωστό και πλέων αποδεκτό ότι η απομακρυσμένη φροντίδα ενός χωραφιού είναι πρακτικά αδύνατη λόγω της έλλειψης του απαραίτητου εξοπλισμού που θα επιτρέψει τον εκ των πραγμάτων ολικό έλεγχο, χωρίς να προϋποτίθενται εξεζητημένες παροχές (πχ. Επίγειο internet) και συνάμα τη συνολική οικονομία σε νερό εφόσον θα μπορεί ή πλατφόρμα από μόνη της να πάρει αποφάσεις ώστε να διακόψει το πότισμα εφόσον δεν χρειάζεται .

Η παρούσα εργασία, υλοποιεί μία πλατφόρμα η οποία θα λύνει το πρόβλημα της απομακρυσμένης φροντίδας ενός χωραφιού ή οποιουδήποτε χώρου που χρίζει ποτίσματος. Μέχρι τώρα οι υφιστάμενες λύσεις που υπάρχουν στην αγορά, είναι μικρής κλίμακας , ή στερούν ευφυΐας, ή δεν μπορούν να δώσουν «εικόνα» ή να χειριστούν απομακρυσμένα. Η συγκεκριμένη εργασία, είναι μία προσπάθεια να καλύψουμε πολλές από τις ανάγκες που εγείρονται όσο αφορά τη φροντίδα ενός χωραφιού, με το χαμηλότερο κόστος και τη δυνατότητα επέκτασης.

Σκοπός αυτής της πτυχιακής είναι να κατασκευάσουμε ένα σύστημα το οποίο θα ποτίζει μία ή δύο φορές ημερησίως ανάλογα με το πρόγραμμα που θα προτείνουμε, όμως ανάλογα τις καιρικές συνθήκες θα «αποφασίζει» δυναμικά αν πρέπει να προβεί στην εκάστοτε κίνηση. Δηλαδή, σε μέρες μεγάλης ξηρασίας, θα μπορεί να παρατείνει το χρόνο ποτίσματος αψηφώντας το πρόγραμμά του, ή αντίστροφα σε μέρες που βρέχει θα μπορεί να προσπερνά οποιοδήποτε από τα προγραμματισμένα του ποτίσματα.

Βεβαίως το σύστημα αφενός θα καταγράφει τις κινήσεις του αλλά και αφετέρου, θα ενημερώνει μέσω SMS όταν κάποια τέτοια κίνηση παράβλεψης του χρονοδιαγράμματος έχει λάβει χώρα. Ο δε χρήστης βλέποντας τις μετρήσεις θερμοκρασίας υγρασίας και υγρασίας χώματος, και κρίνοντας βλέποντας πότε έγινε το τελευταίο πότισμα, θα μπορεί να ξεκινήσει το πότισμα ή ενδεχομένως να το σταματήσει.

Ο απομακρυσμένος έλεγχος της πλατφόρμας θα γίνεται μέσω έξυπνου κινητού με μία εφαρμογή Android. Τα δεδομένα θα καταγράφονται ανά τακτά χρονικά διαστήματα σε ένα απομακρυσμένο MySQL server και μέσω αυτού θα ενημερώνεται και ο χρήστης για τα LIVE δεδομένα.

Η πτυχιακή δομείται σε Κεφάλαια ως ακολούθως:

- Στο πρώτο Κεφάλαιο γίνεται εκτενής παρουσίαση στην τεχνολογία των αισθητήρων, όσον αφορά τα είδη και τα χαρακτηριστικά των αισθητήρων, την ιστορική εξέλιξή τους ανά τα χρόνια και τελικά, τις πρακτικές εφαρμογές τους.
- Στο δεύτερο Κεφάλαιο γίνεται μια ανάλογη παρουσίαση όσον αφορά τους μικροελεγκτές, δηλαδή το σημαντικότερο και πιο πολύπλοκο κομμάτι της εργασίας. Επίσης γίνεται αναφορά στους τύπους, την ιστορική εξέλιξη και φυσικά στις πρακτικές εφαρμογές, οι οποίες είναι αμέτρητες.
- Στο τρίτο Κεφάλαιο αναφέρεται επιγραμματικά το πρόβλημα το οποίο καλούμαστε να λύσουμε. Γίνεται αναφορά στις ήδη υπάρχουσες λύσεις τις αγοράς και αντιπαρατίθενται οι λειτουργίες τους με τις απαιτήσεις του προβλήματος. Τέλος αναφέρονται οι δυσκολίες που θα πρέπει να αντιμετωπιστούν ούτως ώστε να επιτευχτεί η βέλτιστη λύση.

- Στο τέταρτο κεφάλαιο προσεγγίζουμε την υλοποίηση του συστήματος αναλύοντας τα χαρακτηριστικά που θα πρέπει να έχει, και επιπρόσθετα αναφέρουμε με ποιο τρόπο ή τμε τι υλικό θα τα υλοποιήσουμε. Φυσικά γίνεται αναφορά όλων των εργαλείων που θα χρησιμοποιηθούν για την περάτωση της εργασίας.
- Στο πέμπτο κεφάλαιο γίνεται περιγραφή και εξήγηση όλων των εξαρτημάτων που θα χρησιμοποιηθούν για την κατασκευή του συστήματος, και περιγράφεται αναλυτικά η συνδεσμολογία του. Δηλαδή περιγράφεται λεπτομερώς ο τρόπος σύνδεσης των εξαρτημάτων με την κεντρική αναπτυξιακή πλακέτα κάθε ένα ξεχωριστά και όλα μαζί.
- Στο έκτο κεφάλαιο αναλύεται και περιγράφεται λεπτομερώς η ο κώδικας που χρησιμοποιήθηκε τόσο για τον προγραμματισμό της αναπτυξιακής πλακέτας με τις περιφερειακές συσκευές που το αποτελούν όσο και τον κώδικα που χρησιμοποιήθηκε για την κατασκευή της android εφαρμογής και του back end του server που θα φιλοξενεί τα καταγεγραμμένα δεδομένα του συστήματος.
- Στο έβδομο κεφάλαιο περιγράφουμε την πρακτική λειτουργία του συστήματος αναλύοντας κάποια σενάρια λειτουργίας και αναλύοντας βήμα-βήμα τις κινήσεις του χρήστη.
- Στο όγδοο και τελευταίο κεφάλαιο της αναφοράς, κάνουμε κάποιες παρατηρήσεις για την συγκεκριμένη προσέγγιση αυτής της υλοποίησης και εξαγάγουμε κάποια συμπεράσματα όσον αφορά το ίδιο το σύστημα και την πιθανή μελλοντική του εξέλιξη.

ΠΕΡΙΕΧΟΜΕΝΑ

ABSTRACT **2**

ΣΥΝΟΨΗ **2**

ΚΙΝΗΤΡΟ ΓΙΑ ΤΗΝ ΔΙΕΞΑΓΩΓΗ ΤΗΣ ΕΡΓΑΣΙΑΣ **3**

ΠΕΡΙΕΧΟΜΕΝΑ **5**

ΠΙΝΑΚΑΣ ΣΧΗΜΑΤΩΝ **9**

ΠΙΝΑΚΑΣ ΠΙΝΑΚΩΝ **12**

1. ΚΕΦΑΛΑΙΟ 1 ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΤΕΧΝΟΛΟΓΙΑ ΤΩΝ ΑΙΣΘΗΤΗΡΩΝ **13**

1.1 Εισαγωγή **13**

1.2 Ορισμός **13**

1.3 Μεταγενέστερη ανάπτυξη αισθητήρων **15**

1.4 Ταξινόμηση Αισθητήρων με Βάση τη Μορφή Σήματος **17**

1.5 Στατικά χαρακτηριστικά των αισθητήρων **18**

1.5.1 Ακρίβεια (accuracy) 18

1.5.2 Ακρίβεια (precision), Επαναληψιμότητα, Αναπαραγωγιμότητα 18

1.5.3 Ανοχή 18

1.5.4 Εύρος 19

1.5.5 Συστηματικό σφάλμα 19

1.5.6 Γραμμική απόκριση 19

1.5.7 Ευαισθησία στη μέτρηση 20

1.5.8 Ευαισθησία στη διαταραχή 20

1.5.9 Υστέρηση 21

1.5.10 Νεκρό εύρος 23

1.5.11 Κατώφλι 23

1.5.12 Διακριτική ικανότητα 23

1.6 Αισθητήρες θερμοκρασίας **23**

1.6.1 Θερμίστορ (thermistor) 24

1.6.2 Θερμοζεύγη (thermocouples) 26

1.6.3 Αισθητήρες θερμικών αντιστάσεων (Resistance Temperature Detector (RTD)) 29

1.7 Αισθητήρες υγρασίας **30**

1.7.1 Εισαγωγή 30

1.7.2 Λειτουργία 30

1.7.3 Υλικά κατασκευής 32

2. ΚΕΦΑΛΑΙΟ 2 ΜΙΚΡΟΕΛΕΓΚΤΕΣ **33**

2.1 Εισαγωγή 33

2.2 Η μνήμη στον μικροελεγκτή 34

2.3 Οι θύρες Επικοινωνίας I/O 35

2.4 Περιφερειακές Μονάδες σε ένα Τυπικό Μικροελεγκτή 35
2.4.1 Μονάδες Timer 36
2.4.2 Η Σειριακή Θύρα 37
2.4.3 Analog to digital Converter (ADC) 38

2.5 Διαδεδομένες Κατηγορίες Μικροελεγκτών 38

2.6 Κατασκευαστές 40

2.7 Δημοφιλή Μοντέλα Μικροελεγκτών 40
2.7.1 Atmel AVR Development Boards 40
2.7.2 Raspberry Pi 41
2.7.3 Arduino Family 42

2.8 Πεδία Εφαρμογών 43
2.8.1 Σύστημα Παρακολούθησης Καιρικών Συνθηκών 43
2.8.2 Τρισδιάστατοι Εκτυπωτές/Ρούτερ CNC 45
2.8.3 Ρομπότ 45
2.8.4 Έξυπνο σπίτι 46
2.8.5 Αυτόματο πότισμα 47

2.9 Χρησιμότητα και Χρηστικότητα 47

3. ΚΕΦΑΛΑΙΟ 3 ΕΥΦΥΕΣ ΣΥΣΤΗΜΑ ΠΟΤΙΣΜΑΤΟΣ ΜΕ ΑΠΟΜΑΚΡΥΣΜΕΝΟ ΕΛΕΓΧΟ 48

3.1 Το Πρόβλημα που Καλούμαστε να Επιλύσουμε 48

3.2 Έρευνα Αγοράς 49
3.2.1 Hunter NODE 100 49
3.2.2 Rain Bird 49
3.2.3 GreenBox 50

3.3 Διαφορές 50

3.4 Προβλήματα που εγείρονται 50

4. ΚΕΦΑΛΑΙΟ 4 ΥΛΟΠΟΙΗΣΗ ΣΥΣΤΗΜΑΤΟΣ 52

4.1 Εξοπλισμός και Εξήγηση 52

4.2 Προγραμματιστικά Εργαλεία 55

4.3 Σχεδιασμός 57
4.3.1 Συνδεσιμότητα 58
4.3.2 Καταγραφή 59
4.3.3 Αλληλεπίδραση 59

5. ΚΕΦΑΛΑΙΟ 5 ΣΥΝΔΕΣΜΟΛΟΓΙΑ 60

5.1 Γενικά 60

5.2 Αναπτυξιακή Πλακέτα 60
5.2.1 Real Time Clock 61
5.2.2 Πλακέτα Ρελέ NO/NC 62
5.2.3 Ηλεκτροβάνες 62
5.2.4 Ρελέ 62
5.2.5 Arduino Relay Module 63

5.3 Αισθητήρες 65
5.3.1 Θερμοκρασία/Υγρασία 65
5.3.2 Υγρασία Χώματος 66

5.4 GSM/GPRS 67

5.5 Διεπαφή/Αποθήκευση 68

5.6 Ολική Συνδεσμολογία Συστήματος 70

6. ΚΕΦΑΛΑΙΟ 6 ΥΛΟΠΟΙΗΣΗ ΛΟΓΙΣΜΙΚΟΥ 71

6.1 Προγραμματισμός Ελεγκτή 71
6.1.1 Λειτουργία του Προγράμματος 71
6.1.2 Σειριακή Διεπαφή 72
6.1.3 GSM/GPRS 72
6.1.4 SdCard 73

6.2 Αρχικοποίηση Συστήματος 74
6.2.1 Read File 74
6.2.2 InitSchedule 76
6.2.3 Πότισμα 77
6.2.4 Έλεγχος SMS 78
6.2.5 SQL Post 81
6.2.6 Θερμοκρασία/Υγρασία Περιβάλλοντος 82
6.2.7 Οθόνη αφής 82

6.3 Μενού 83

6.4 Δημοσίευση Δεδομένων 95
6.4.1 BackEnd 96
6.4.2 FrontEnd 98

6.5 Android 99
6.5.1 Προγραμματιστικά Εργαλεία 99
6.5.2 Προγραμματισμός 100
6.5.3 Η Εφαρμογή 100
6.5.4 Ο Κώδικας 101
6.5.5 Ενέργειες 103
6.5.6 Permissions 106

7. ΚΕΦΑΛΑΙΟ 7 ΣΥΝΟΛΙΚΗ ΛΕΙΤΟΥΡΓΙΑ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ 107

7.1 Σενάριο 1 Χειροκίνητη Έναρξη Ποτίσματος 107
7.2 Σενάριο 2 Αλλαγή στο Πρόγραμμα Ποτίσματος 108
7.3 Σενάριο 3 Απομακρυσμένος έλεγχος 110
8. ΚΕΦΑΛΑΙΟ 8 ΣΥΜΠΕΡΑΣΜΑΤΑ 112
ΒΙΒΛΙΟΓΡΑΦΙΑ 113
ΠΑΡΑΡΤΗΜΑ 114

ΠΙΝΑΚΑΣ ΣΧΗΜΑΤΩΝ

ΚΕΦΑΛΑΙΟ 1

ΣΧΗΜΑ 1-1 ΠΑΡΑΔΕΙΓΜΑ ΑΙΣΘΗΤΗΡΑ ΣΧΕΣΕΩΝ ΕΙΣΟΔΟΥ/ΕΞΟΔΟΥ 14
ΣΧΗΜΑ 1-2 ΠΟΡΕΙΑ ΤΩΝ ΑΙΣΘΗΤΗΡΩΝ ΣΤΑ ΑΥΤΟΚΙΝΗΤΑ ΜΕ ΤΗΝ ΠΑΡΟΔΟ ΤΟΥ ΧΡΟΝΟΥ 14
ΣΧΗΜΑ 1-3 ΥΛΙΚΑ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑ ΕΠΕΞΕΡΓΑΣΙΑΣ ΔΗΜΙΟΥΡΓΟΥΝ ΤΟΥΣ ΑΙΣΘΗΤΗΡΕΣ 15
ΣΧΗΜΑ 1-4 ΠΑΡΑΔΕΙΓΜΑ ΤΟΥ ΑΝΘΡΩΠΙΝΟΥ ΣΥΣΤΗΜΑΤΟΣ ΑΙΣΘΗΣΗΣ 16
ΣΧΗΜΑ 1-5 ΠΡΟΩΡΟΣ ΟΠΤΙΚΟΣ ΜΕΤΑΤΡΟΠΕΑΣ 17
ΣΧΗΜΑ 1-6 ΧΑΡΑΚΤΗΡΙΣΤΙΚΗ ΕΞΟΔΟΥ ΑΙΣΘΗΤΗΡΑ 19
ΣΧΗΜΑ 1-7 ΟΛΙΣΘΗΣΗ ΤΟΥ ΜΗΔΕΝΟΣ, (Β) ΟΛΙΣΘΗΣΗ ΕΥΑΙΣΘΗΣΙΑΣ, (C) ΣΥΝΔΥΑΣΜΕΝΗ ΕΠΙΔΡΑΣΗ ΤΩΝ ΔΥΟ ΟΛΙΣΘΗΣΕΩΝ 20
ΣΧΗΜΑ 1-8 ΥΣΤΕΡΗΣΗ ΕΞΟΔΟΥ 22
ΣΧΗΜΑ 1-9 ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΕΞΟΔΟΥ ΑΙΣΘΗΤΗΡΑ ΜΕ ΝΕΚΡΟ ΕΥΡΟΣ 22
ΣΧΗΜΑ 1-10 Α) ΑΝΤΙΣΤΑΣΕΙΣ ΑΝΙΧΝΕΥΣΗΣ ΘΕΡΜΟΚΡΑΣΙΑΣ (RTDS) Β) ΘΕΡΜΙΣΤΟΡ 24
ΣΧΗΜΑ 1-11 ΔΙΑΦΟΡΑ ΣΧΗΜΑΤΑ ΘΕΡΜΙΣΤΟΡ 24
ΣΧΗΜΑ 1-12 ΤΥΠΟΙ ΘΕΡΜΟΖΕΥΓΩΝ ΙΔΙΟΤΗΤΕΣ ΤΩΝ ΘΕΡΜΟΖΕΥΓΩΝ 26
ΣΧΗΜΑ 1-13 ΑΙΣΘΗΤΗΡΕΣ ΘΕΡΜΙΚΩΝ ΑΝΤΙΣΤΑΣΕΩΝ (RTDS) 29
ΣΧΗΜΑ 1-14 ΔΙΑΓΡΑΜΜΑ ΤΩΝ ΑΣΥΡΜΑΤΩΝ ΑΙΣΘΗΤΗΡΩΝ SAW, ΓΙΑ ΤΑΥΤΟΧΡΟΝΗ ΜΕΤΡΗΣΗ ΘΕΡΜΟΚΡΑΣΙΑΣ ΚΑΙ ΥΓΡΑΣΙΑΣ ΓΙΑ 915 MHZ 31
ΣΧΗΜΑ 1-15 (Α) ΜΕΤΡΗΣΗ ΥΓΡΑΣΙΑΣ 32

ΚΕΦΑΛΑΙΟ 2

ΣΧΗΜΑ 2-1 ΚΥΡΙΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΕΝΟΣ ΜΙΚΡΟΕΛΕΓΚΤΗ 33
ΣΧΗΜΑ 2-2 ΈΝΑΣ ΜΙΚΡΟΕΛΕΓΚΤΗΣ ΜΟΝΟΥ CHIP. 34
ΣΧΗΜΑ 2-3 Η ROM ΕΝΟΣ ΜΙΚΡΟΕΛΕΓΚΤΗ 34
ΣΧΗΜΑ 2-4 Η RAM ΕΝΟΣ ΜΙΚΡΟΕΛΕΓΚΤΗ 35
ΣΧΗΜΑ 2-5 ΤΥΠΙΚΟ ΠΑΡΑΔΕΙΓΜΑ ΔΙΕΠΑΦΗΣ 35
ΣΧΗΜΑ 2-6 ΠΑΡΑΔΕΙΓΜΑ ΕΞΟΔΟΥ ΤΟΥ TIMER 36
ΣΧΗΜΑ 2-7 ΠΑΡΑΔΕΙΓΜΑ ΕΙΣΟΔΟΥ ΣΤΟΝ TIMER 36
ΣΧΗΜΑ 2-8 Η ΣΕΙΡΙΑΚΗ ΘΥΡΑ 37
ΣΧΗΜΑ 2-9 ATMEL 1244 8-BIT AVR MICROCONTROLLER 38
ΣΧΗΜΑ 2-10 16-BIT NEC MICROCONTROLLER 39
ΣΧΗΜΑ 2-11 32-BIT AVR MICROCONTROLLER 39
ΣΧΗΜΑ 2-12 ATMEL AVR DEVELOPMENT BOARD 41
ΣΧΗΜΑ 2-13 RASPBERRY PI B/B+ 41
ΣΧΗΜΑ 2-14 ARDUINO UNO 42
ΣΧΗΜΑ 2-15 ARDUINO MEGA 42
ΣΧΗΜΑ 2-16 ARDUINO PRO MINI BASED WEATHER STATION 44
ΣΧΗΜΑ 2-17 3D CNC PRINTER 45
ΣΧΗΜΑ 2-18 ARDUINO ROBOT 45
ΣΧΗΜΑ 2-19 ARDUINO SMART HOME 46
ΣΧΗΜΑ 2-20 FEZ SMART HOME SYSTEM 46
ΣΧΗΜΑ 2-21 ARDUINO AUTOMATIC PLANT WATERING 47

ΚΕΦΑΛΑΙΟ 3

ΣΧΗΜΑ 3-1 ΤΟ ΠΡΟΣ ΥΛΟΠΟΙΗΣΗ ΣΥΣΤΗΜΑ 48
ΣΧΗΜΑ 3-2 HUNTER NODE 49
ΣΧΗΜΑ 3-3 RAIN BIRD 49
ΣΧΗΜΑ 3-4 GREENBOX 50

ΚΕΦΑΛΑΙΟ 4

ΣΧΗΜΑ 4-1 GPRS SHIELD 52
ΣΧΗΜΑ 4-2 ARDUINO COMPATIBLE SOIL MOISTURE SENSOR / TEMPERATURE-HUMIDITY SENSOR DHT11 53
ΣΧΗΜΑ 4-3 ARDUINO COMPATIBLE RTC (REAL TIME CLOCK) MODULE 53
ΣΧΗΜΑ 4-4 ARDUINO COMPATIBLE RELAY MODULE 54
ΣΧΗΜΑ 4-512V ELECTROVALVE 54
ΣΧΗΜΑ 4-63.2" TOUCH SCREEN MODULE 54
ΣΧΗΜΑ 4-7 TFT SD CARD SHIELD FOR ARDUINO 55
ΣΧΗΜΑ 4-8 ARDUINO OPEN SOURCE COMMUNITY 56
ΣΧΗΜΑ 4-9 THE ECLIPSE PROJECT 57
ΣΧΗΜΑ 4-10 PROJECT PLAN 58

ΚΕΦΑΛΑΙΟ 5

ΣΧΗΜΑ 5-1 ΣΥΝΔΕΣΗ ARDUINO ΜΕ RTC 61
ΣΧΗΜΑ 5-2 ΗΛΕΚΤΡΟΒΑΝΑ 62
ΣΧΗΜΑ 5-3 RELAY TYPES 63
ΣΧΗΜΑ 5-4 RELAY MODULE 63
ΣΧΗΜΑ 5-5 ΣΥΝΔΕΣΗ ARDUINO ΜΕ RELAY MODULE ΚΑΙ ΗΛΕΚΤΡΟΒΑΝΑΣ 64
ΣΧΗΜΑ 5-6 ΑΙΣΘΗΤΗΡΑΣ DHT11 65
ΣΧΗΜΑ 5-7 ΣΥΝΔΕΣΗ ARDUINO ΜΕ DHT11 66
ΣΧΗΜΑ 5-8 ΣΥΝΔΕΣΗ ARDUINO ΜΕ ΑΙΣΘΗΤΗΡΑ ΥΓΡΑΣΙΑΣ ΧΩΜΑΤΟΣ 67
ΣΧΗΜΑ 5-9 ΣΥΝΔΕΣΗ ARDUINO ΜΕ GPRS MODULE 67
ΣΧΗΜΑ 5-10 TFT PINOUT 68
ΣΧΗΜΑ 5-11ΟΛΙΚΗ ΣΥΝΔΕΣΜΟΛΟΓΙΑ ΣΥΣΤΗΜΑΤΟΣ 70

ΚΕΦΑΛΑΙΟ 6

ΣΧΗΜΑ 6-1 MAIN SCREEN VIEW 85
ΣΧΗΜΑ 6-2 MAINMANU VIEW 87
ΣΧΗΜΑ 6-3 VALVECONTROL VIEW 88
ΣΧΗΜΑ 6-4 PARAMS VIEW 89
ΣΧΗΜΑ 6-5 RELAYS VIEW 90
ΣΧΗΜΑ 6-6 WATERINGS VIEW 92
ΣΧΗΜΑ 6-7 EDIT_SCHEDULE VIEW 93
ΣΧΗΜΑ 6-8 SCHEDULES VIEW 94
ΣΧΗΜΑ 6-9 WAMP SERVER 95
ΣΧΗΜΑ 6-10 JSON 98
ΣΧΗΜΑ 6-11 APACHE FRONTEND 98
ΣΧΗΜΑ 6-12 PHPMYADMIN 99
ΣΧΗΜΑ 6-13 ECLIPSE WORKSPACE 100
ΣΧΗΜΑ 6-14 EASY GARDENER 101
ΣΧΗΜΑ 6-15 MAIN LAYOUT DESIGN 102

ΚΕΦΑΛΑΙΟ 7

ΣΧΗΜΑ 7-7-1 ΚΕΝΤΡΙΚΗ ΟΘΟΝΗ 107
ΣΧΗΜΑ 7-7-2 MAIN MENU 107
ΣΧΗΜΑ 7-7-3 CONTROL SCREEN 108
ΣΧΗΜΑ 7-7-4 CONTROL SCREEN 2REALAY BOARD 108
ΣΧΗΜΑ 7-7-5 ΡΥΘΜΙΣΕΙΣ 109
ΣΧΗΜΑ 7-7-6 SCHEDULE SCREEN 109
ΣΧΗΜΑ 7-7-7 EDIT SCHEDULE 1 SCREEN 109
ΣΧΗΜΑ 7-8 EASY GARDENER INTERFACE 110
ΣΧΗΜΑ 7-9 SYNC PRESSED 110
ΣΧΗΜΑ 7-10 SEND OK MESSAGE 111
ΣΧΗΜΑ 7-11 MAIN SCREEN ALTERED 111

ΠΙΝΑΚΑΣ ΠΙΝΑΚΩΝ

ΚΕΦΑΛΑΙΟ 1
ΠΙΝΑΚΑΣ 1-1 ΤΑΞΙΝΟΜΗΣΗ ΑΙΣΘΗΤΗΡΩΝ 17
ΠΙΝΑΚΑΣ 1-2 ΣΕΙΡΑ ΤΥΠΟΠΟΙΗΜΕΝΩΝ ΕΙΔΩΝ ΤΩΝ ΑΙΣΘΗΤΗΡΩΝ ΘΕΡΜΟΖΕΥΓΩΝ 28
ΠΙΝΑΚΑΣ 1-3 ΚΟΙΝΑ ΜΕΤΑΛΛΑ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΟΥΝΤΑΙ ΣΕ RTDS 29
ΚΕΦΑΛΑΙΟ 5
ΠΙΝΑΚΑΣ 5-1 TFT PINOUT 1 69
ΠΙΝΑΚΑΣ 5-2 TFT PINOUT 2 69
ΚΕΦΑΛΑΙΟ 6
ΠΙΝΑΚΑΣ 6-1 SENSORS 96

1. ΚΕΦΑΛΑΙΟ 1
ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΤΕΧΝΟΛΟΓΙΑ ΤΩΝ ΑΙΣΘΗΤΗΡΩΝ

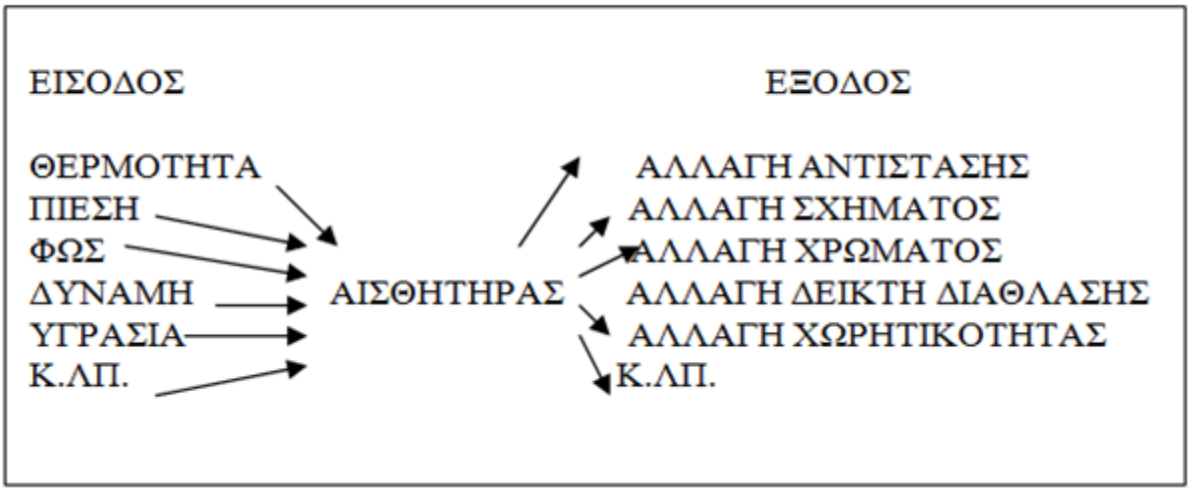
1.1 Εισαγωγή

Η σημασία των αισθητήρων για τον άνθρωπο είναι σχεδόν αυτονόητη. Οι πρώτοι αισθητήρες εμφανίζονται μαζί με τα έμβια όντα και αποτελούν όργανα τους. Το μάτι και το αφτί είναι χαρακτηριστικά παραδείγματα: το πρώτο ανιχνεύει τμήμα του φάσματος της ηλεκτρομαγνητικής ακτινοβολίας και το δεύτερο τον ήχο, δηλαδή κύματα πίεσης. Πολύ αργότερα, ο άνθρωπος συνειδητοποιεί ότι χρειάζεται όργανα μέτρησης για να λύσει καθημερινά πρακτικά προβλήματα, όπως αυτό της μέτρησης του μήκους, του βάρους ή του όγκου. Στη συνέχεια η επιθυμία του ανθρώπου να γνωρίσει τη Φύση αλλά και διάφοροι πρακτικοί λόγοι, δημιουργούν την ανάγκη μέτρησης περισσότερων φυσικών μεγεθών. Ενδεικτικά αναφέρουμε ότι το πρώτο θερμόμετρο εμφανίζεται το 1585, ενώ το βαρόμετρο το 1643.

1.2 Ορισμός

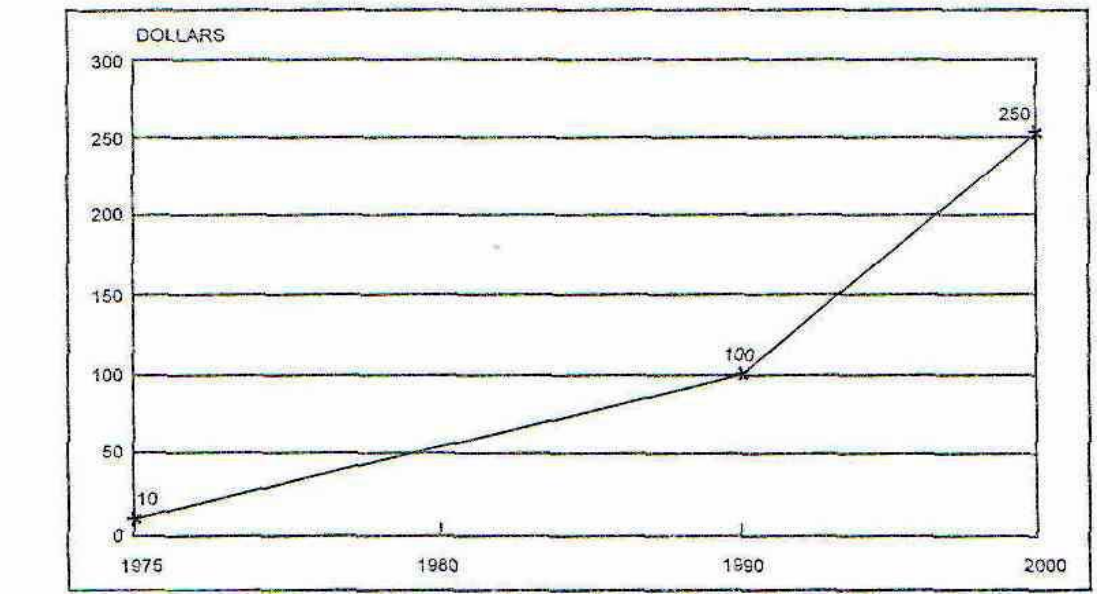
Ο καθορισμός του όρου «αισθητήρα» είναι απλός: Αισθητήρας είναι μια συσκευή ή η διάταξη η οποία ενσωματώνεται πάνω σε κάποια μηχανή επεξεργασίας ενός προϊόντος για την ανίχνευση, την καταγραφή, τη μέτρηση και τη μεταφορά στη μονάδα επεξεργασίας δεδομένων όλων των πληροφοριών που έχουν σχέση με τη κατάσταση λειτουργίας του ελεγχόμενου συστήματος. Οι αισθητήρες μετατρέπουν ένα φυσικό μέγεθος (φωτεινό σήμα, μηχανικό σήμα, υγρασία, πίεση κ.λπ.) σε ηλεκτρικό σήμα, ή σε υδραυλική πίεση κτλ. Χωρίς αισθητήρες, δεν υπάρχει κανένα δεδομένο για τον έλεγχο της θερμοκρασία στο σπίτι, το σύστημα ελέγχου που βοήθησε να προσγειωθεί ένα διαστημικό σκάφος στο φεγγάρι, το σύστημα ανάφλεξης στο αυτοκίνητο, το φως που λειτουργεί στο ψυγείο, ή το σύστημα εξαερισμού του κτιρίου στο οποίο εργαζόμαστε.

Οι αισθητήρες είναι, κυριολεκτικά, οι «αισθήσεις» των συστημάτων ελέγχου (ένα σύστημα ελέγχου ορίζεται ως ένας βρόχος ελέγχου είσοδος/απόφαση/έξοδος - μπορεί να είναι τόσο απλός όσο ένα on-off κύκλωμα ή τόσο σύνθετος όσο ένα «ενσωματωμένο κτίριο»).



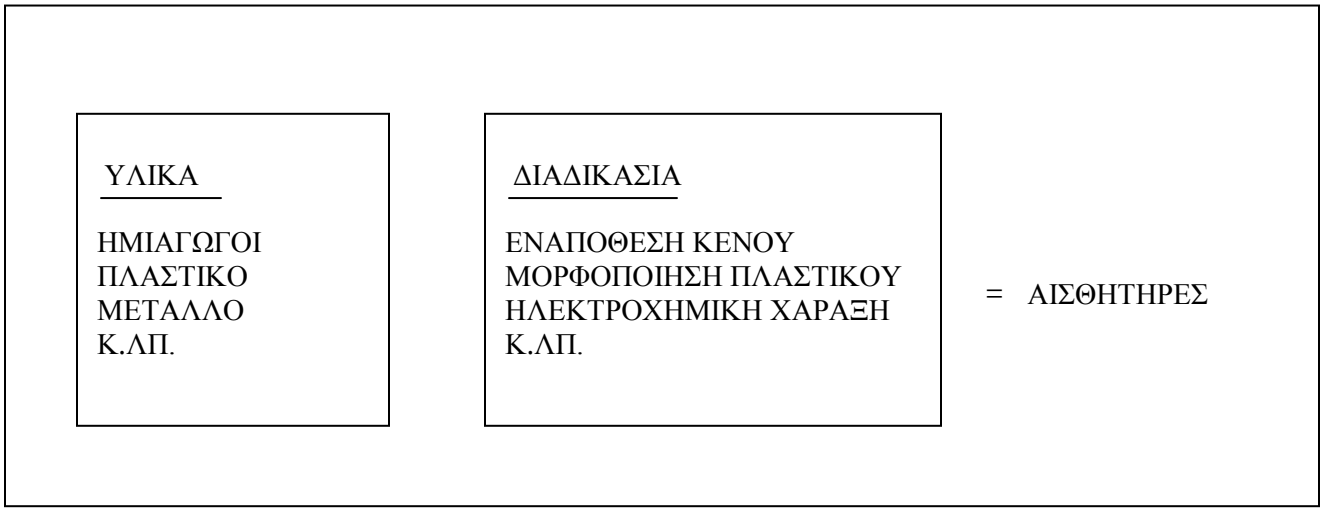
Σχήμα 1-1 Παράδειγμα Αισθητήρα σχέσεων Εισόδου/Εξόδου

Ένα παράδειγμα με το οποίο μπορούμε όλοι να αναγνωρίσουμε είναι το πλήθος των αισθητήρων που βρίσκεται τώρα στα αυτοκίνητα (Σχήμα 1.2). Αυτό ανάγκασε τους αυτοκινητιστές να προσθέσουν περισσότερους βρόχους ελέγχου - και, επομένως, περισσότερους αισθητήρες - στα αυτοκινητικά σχέδιά τους, με συνέπεια να υπάρξει μια ταχεία ανάπτυξη της ηλεκτρονικής περιεκτικότητας σε αισθητήρες



Σχήμα 1-2 πορεία των αισθητήρων στα αυτοκίνητα με την πάροδο του χρόνου

Είναι επίσης σημαντικό να κατανοήσουμε τη διαφορά μεταξύ "των αισθητήρων" και της "τεχνολογίας αισθητήρων". Ο αισθητήρας είναι η συσκευή που συλλέγει και διαβιβάζει τα δεδομένα εισόδου - δεν υπάρχει κανένα τέτοιο πράγμα ως τεχνολογία αισθητήρων αυτό καθ' εαυτό. Η τεχνολογία αισθητήρων στην πραγματικότητα είναι μια συσσώρευση των υλικών και οι τεχνικές επεξεργασίας που λειτουργούν μαζί για να παρέχουν την απαραίτητη λειτουργία στο σύστημα ελέγχου εφαρμογών, εφαρμογές συλλογής στοιχείων, και εφαρμογές ελέγχου / ώθησης

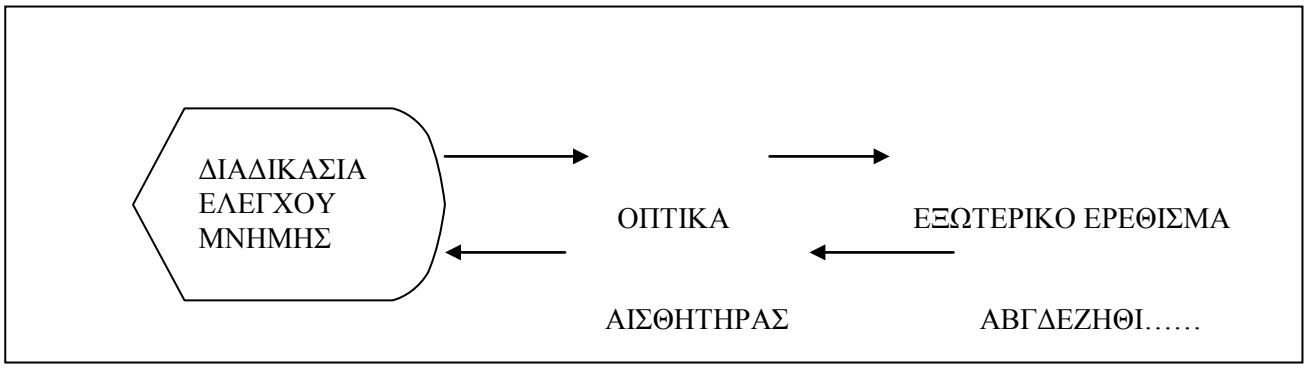


Σχήμα 1-3 Υλικά και τεχνολογία επεξεργασίας δημιουργούν τους αισθητήρες

1.3 Μεταγενέστερη ανάπτυξη αισθητήρων

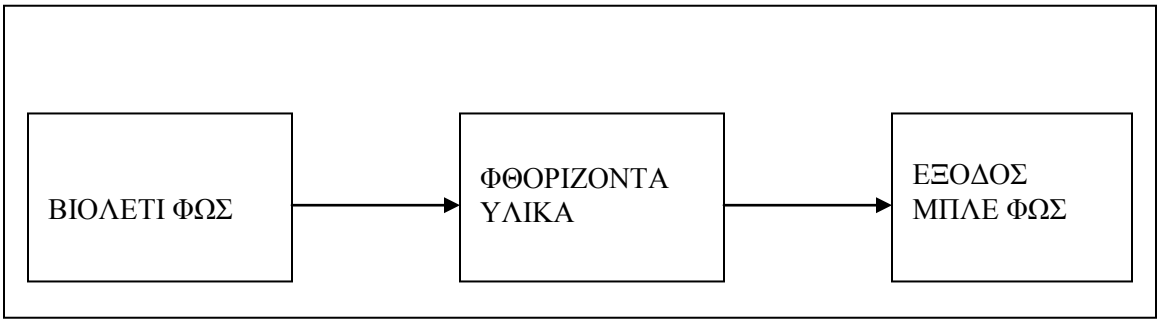
Παραδείγματα αίσθησης και μετατροπής

Η ιδέα "της αίσθησης" είναι τόσο παλαιά όσο η ζωή η ίδια. Συναντάμε "την αίσθηση" κάθε στιγμή της ημέρας : Τα αισθητήρια όργανά μας αλληλεπιδρούν με το περιβάλλον μας και μετασχηματίζουν την αισθητήρια είσοδο σε ερεθίσματα που μας αναγκάζει να ενεργούμε. Αυτή η είσοδος μπορεί να μας προειδοποιήσει για τον κίνδυνο ή η ταλαιπωρία, ή μας επιτρέπει να εκτελέσουμε μια δραστηριότητα, όπως η ανάγνωση ενός βιβλίου (βλ. Σχήμα 1.4).
Το σώμα μας χρειάζεται έναν οπτικό αισθητήρα για να μετατρέψει τις πληροφορίες αναγνώρισης σχεδίων (τυπωμένες λέξεις σε μια σελίδα) στον εγκέφαλο, όπου αποθηκεύεται και οι αποφάσεις γίνονται βασισμένες σε εκείνη την είσοδο. Ενδεχομένως - εάν, παραδείγματος χάριν, πρέπει να είμαστε σε θέση να φέρουμε στην μνήμη μας αυτό το υλικό για μια εξέταση – αυτή η είσοδος γίνεται έπειτα ένα μέρος της μνήμης του "συστήματος ελέγχου " του ανθρώπου.



Σχήμα 1-4 Παράδειγμα του ανθρώπινου συστήματος αίσθησης

Οι άνθρωποι, φυσικά, είναι το τελευταίο σύστημα ελέγχου - είναι ουσιαστικά εύκολο για μας να διαβάσουμε τις λέξεις σε αυτήν την σελίδα. Εντούτοις, δεδομένου ότι οι κατασκευαστές και οι επιστήμονες άρχισαν τη διαδικασία προσδιορισμού των αρχικών αναγκών για τους αισθητήρες, έγινε προφανές ότι η αντίληψη επρόκειτο να είναι σύνθετη. Ακριβώς σκεφτόμαστε την έκταση της πολυπλοκότητας που περιλαμβάνεται σε μια μηχανή για να διαφοροποιήσει κάτι τόσο απλό όσο το χρώμα (παραδείγματος χάριν, διαφοροποιεί τα κόκκινα μέρη από τα μπλε μέρη και έπειτα ταξινομεί εκείνα τα μέρη αναλόγως). Η "απλή" πράξη της ανάγνωσης θα περιλάμβανε αμέτρητες ώρες προγραμματισμό του υπολογιστή και πολύ περίπλοκες ικανότητες αντίληψης για να επιτρέψουν μια οπτική συσκευή αρχικά να διαφοροποιήσει το ένα γράμμα από το άλλο και έπειτα μια λέξη από μια άλλη και μετά να "καταλάβει" την έννοια μιας απλής φράσης ή μιας πρότασης. Οι πιο πρόωροι τύποι αισθητήρων ήταν πραγματικά φυσικές εκδηλώσεις των φυσικών γεγονότων ή των αλλαγών στα υπάρχοντα υλικά που θα μπορούσαν "να γίνουν αισθητά" αλλά "να μην ελεγχθούν." Παραδείγματος χάριν, όταν παγώνει το νερό διαμορφώνεται στον πάγο. Ο πάγος είναι μια διαφορετική μορφή νερού και, από μία άποψη, αντιπροσωπεύει έναν τύπο αισθητήρα θερμοκρασίας επειδή αλλάζει τη μορφή του σε μια πολύ προβλέψιμη, συγκεκριμένη θερμοκρασία. Φυσικά, έγινε σύντομα σημαντικό όχι μόνο να γίνονται αισθητές οι λειτουργίες αλλά και να τις ελέγχουμε - κατά συνέπεια, γεννήθηκαν οι πρώτοι αισθητήρες.Άρχισαν να γίνονται ανακαλύψεις. Αρχίσαμε να μαθαίνουμε για τη μετατροπή, μέσω τέτοιων παραδειγμάτων, όπως τον φθορισμό όταν παρατηρήσαμε ότι μερικά φυσικά υλικά (που ήταν γνωστά αργότερα ως "φθορίζοντα") - κάτω από ορισμένη διέγερση - θα εξέπεμπαν φως ενός μήκους κύματος διαφορετικού από το φως που έλαμπε σε αυτά. Ένα άλλο παράδειγμα ήταν η ανακάλυψη ότι η ανθρώπινη τρίχα θα άλλαζε το φυσικό μήκος της ανάλογα με τη σχετική υγρασία στο περιβάλλον. Αυτό το φαινόμενο έχει χρησιμοποιηθεί από τότε με άλλα υλικά (όπως το νάιλον) για να δημιουργήσουμε μια «οικογένεια» χαμηλού κόστους, εμπορικά διαθέσιμων αισθητήρων υγρασίας.



Σχήμα 1-5 Πρόωρος οπτικός μετατροπέας

1.4 Ταξινόμηση Αισθητήρων με Βάση τη Μορφή Σήματος

Είναι αναγκαίο, όταν συζητάμε το θέμα των αισθητήρων, να αποφασίζουμε εάν θα τους ταξινομήσουμε σύμφωνα με τη λειτουργία που επιτελούν (μέτρηση πίεσης, θερμοκρασίας κλπ) ή τη φυσική αρχή στην οποία στηρίζεται η λειτουργία τους (μαγνητική αντίσταση, οπτικά ηλεκτρονικά κτλ). Στην εποχή μας είναι κοινή τακτική να ταξινομούνται οι αισθητήρες με βάση την κύρια μορφή ενέργειας που μεταφέρει το σήμα τους. Ο παρακάτω πίνακας παρουσιάζει τις διάφορες μορφές ενέργειας που συνήθως χρησιμοποιούνται στην ταξινόμηση των αισθητήρων, μαζί με κάποια από τα χαρακτηριστικά μετρούμενα μεγέθη (measurand).

Πίνακας 1-1 Ταξινόμηση Αισθητήρων

Μορφή Σήματος	**Μετρήσιμες Ποσότητες**
Θερμική	Θερμοκρασία, θερμότητα, ροή θερμότητας, εντροπία, θερμική χωριτητκότητα κλπ.
Ακτινοβολία	Ακτίνες-γ, ακτίνες-χ, υπεριώδες, ορατό(φως), υπέρυθρη, μικροκύματα, ραδιοφωνικά κύματα κλπ.
Μηχανική	Μετατόπιση, ταχύτητα, επιτάχυνση, δύναμη, ροπή στρέψης, πίεση, μάζα, ροή κλπ.
Μαγνητική	Μαγνητικό πεδίο, μαγνητική ροή, μαγνητική ροπή, μαγνήτιση, μαγνητική διαπερατότητα κλπ.
Χημική	Υγρασία, pH, συγκέντρωση αερίων και ατμών, τοξικά και εύφλεκτα υλικά, ρυπαντές κλπ.
Βιολογική	Σάκχαρα, πρωτείνες, ορμόνες, αντιγόνα κλπ.
Ηλεκτρική	Φορτίο, ένταση, τάση, αντίσταση, αγωγιμότητα, χωρητικότητα, επαγωγή, διηλεκτρική σταθερά, πολωση, συχνότητα κλπ.

1.5 Στατικά χαρακτηριστικά των αισθητήρων

Τα στατικά χαρακτηριστικά των αισθητήρων αναφέρονται στην κατάσταση κατά την οποία έχει επέλθει ισορροπία μεταξύ αισθητήρα και μετρούμενου μεγέθους. Για να επιτευχθεί κάτι τέτοιο πρέπει το μετρούμενο μέγεθος είτε να είναι σταθερό, είτε να μεταβάλλεται πολύ αργά σε σχέση με τη δυνατότητα του αισθητήρα να αντιληφθεί τη μεταβολή αυτή.

1.5.1 Ακρίβεια (accuracy)

Με τον όρο ακρίβεια αποδίδεται ο αγγλικός όρος accuracy. Η ακρίβεια δεν σχετίζεται με τον αριθμό των δεκαδικών ψηφίων με τον οποίο μπορεί να γίνει η μέτρηση, αλλά με το κατά πόσο το αποτέλεσμα που δίνει ο αισθητήρας πλησιάζει την φυσική πραγματικότητα, μέσα σε ένα λογικό εύρος τιμών. Η ακρίβεια δίνεται συνήθως ως ποσοστό επί του εύρους λειτουργίας του αισθητήρα. Αν, για παράδειγμα, ένας αισθητήρας πίεσης, περιοχής λειτουργίας 0-10 bar έχει ακρίβεια ±1.0% της πλήρους κλίμακας, τότε η μέγιστη
αβεβαιότητα του αισθητήρα είναι ίση με 0,1 bar. Αυτό σημαίνει ότι όταν ο αισθητήρας δίνει ως αποτέλεσμα 1 bar, τότε η μέγιστη αναμενόμενη αβεβαιότητα είναι ίση με το 10% της τιμής αυτής. Για τον λόγο αυτό θα πρέπει το εύρος λειτουργίας των αισθητήρων να είναι όσο το δυνατόν εγγύτερα στο εύρος των μετρούμενων τιμών, ώστε να εξασφαλίζεται η μέγιστη δυνατή ακρίβεια των μετρήσεων. Αν δηλαδή έχουμε μία εφαρμογή στην οποία οι πιέσεις μεταβάλλονται στο διάστημα 0-1 bar είναι λάθος να επιλέξουμε αισθητήρα περιοχής λειτουργίας 0 -10 bar.

1.5.2 Ακρίβεια (precision), Επαναληψιμότητα, Αναπαραγωγιμότητα

Ο όρος ακρίβεια (precision) εκφράζει τον βαθμό ελευθερίας του αισθητήρα από τυχαία σφάλματα. Αν πάρουμε μεγάλο αριθμό μετρήσεων από έναν μεγάλης ακρίβειας αισθητήρα, τότε η μεταξύ τους διασπορά θα είναι μικρή. Η ακρίβεια συγχέεται συχνά με την πιστότητα. Η μεγάλη ακρίβεια δεν σημαίνει κατ' ανάγκην και μεγάλη πιστότητα. Ένας μεγάλης ακρίβειας αισθητήρας μπορεί να έχει κακή πιστότητα. Κακής πιστότητας μετρήσεις από έναν μεγάλης ακρίβειας αισθητήρα, σημαίνει ότι η μετρήσεις έχουν συστηματικό σφάλμα (bias), γεγονός το οποίο μπορεί να διορθωθεί με βαθμονόμηση (διακρίβωση) του αισθητήρα. Οι όροι επαναληψιμότητα και αναπαραγωγιμότητα είναι ταυτόσημοι, χρησιμοποιούνται όμως ο καθένας σε διαφορετικές περιπτώσεις. Και οι δύο αναφέρονται στο πόσο κοντά είναι τα αποτελέσματα ενός αισθητήρα που μετρά το ίδιο σταθερό μέγεθος, ή η μεν επαναληψιμότητα όταν οι συνθήκες μέτρησης είναι σταθερές, η δε αναπαραγωγιμότητα, όταν οι συνθήκες μέτρησης μεταβάλλονται.

1.5.3 Ανοχή

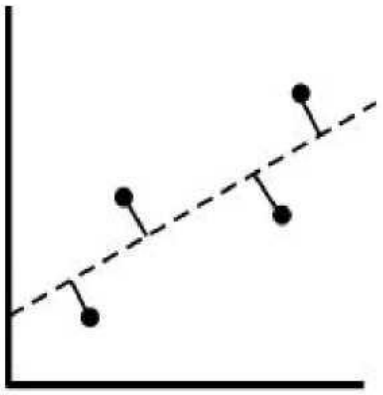
Η ανοχή συνδέεται στενά με την πιστότητα και ορίζει το μέγιστο αναμενόμενο σφάλμα μιας τιμής. Δεν πρόκειται για στατικό χαρακτηριστικό ενός αισθητήρα, αλλά το αναφέρουμε, γιατί πολλές φορές η πιστότητα δίνεται ως ανοχή.

1.5.4 Εύρος

Με τον όρο εύρος αναφερόμαστε στην ελάχιστη και την μέγιστη τιμή του φυσικού μεγέθους που μπορεί να μετρήσει ένας αισθητήρας.

1.5.5 Συστηματικό σφάλμα

Συστηματικό σφάλμα (bias) είναι ένα σταθερό σφάλμα, το ίδιο για όλο το εύρος του αισθητήρα, το οποίο συνήθως μπορεί να μηδενιστεί μέσω βαθμονόμησης.



Σχήμα 1-6 Χαρακτηριστική εξόδου αισθητήρα

Χαρακτηριστικό παράδειγμα συστηματικού σφάλματος εμφανίζεται στις οικιακές ζυγαριές, οι οποίες μπορεί να δείχνουν μη μηδενική ένδειξη, ακόμη και χωρίς φορτίο. Αυτή η μη μηδενική ένδειξη αποτελεί το συστηματικό σφάλμα το οποίο πρέπει να αφαιρέσουμε από την ένδειξη που παίρνουμε κατά τη μέτρηση ώστε να προκύψει η πραγματική τιμή.

1.5.6 Γραμμική απόκριση

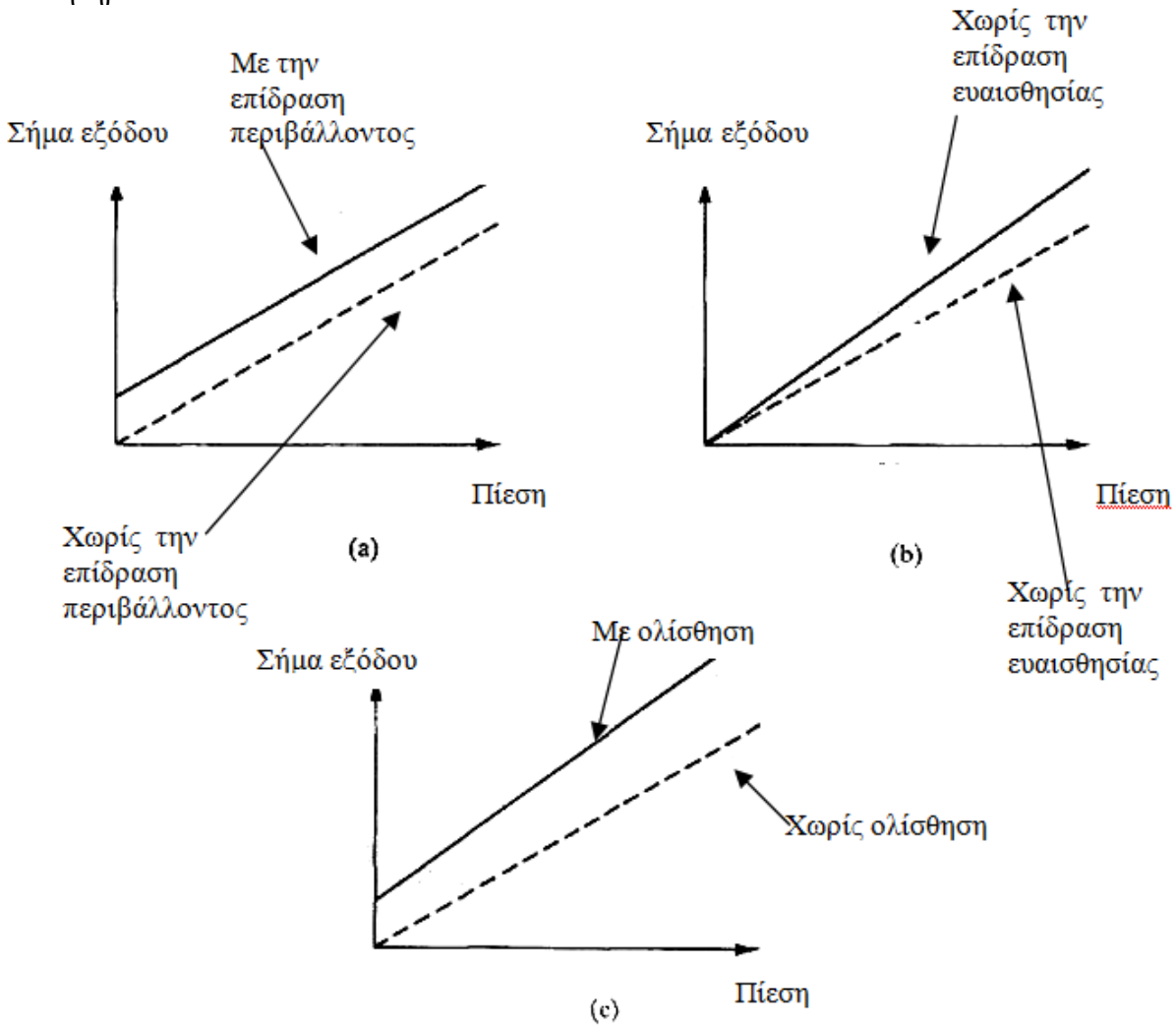
Είναι γενικά επιθυμητό η απόκριση ενός αισθητήρα να μεταβάλλεται γραμμικά με το μετρούμενο μέγεθος. Τα σημεία του Σχήματος 1.6 απεικονίζουν την σχέση μεταξύ σήματος εισόδου (οριζόντιος άξονας) και εξόδου (κατακόρυφος άξονας) ενός αισθητήρα. Η γραμμή μεταξύ των σημείων χαράσσεται εφαρμόζοντας την μέθοδο των ελαχίστων τετραγώνων. Η μη γραμμικότητα εκφράζεται ως η μέγιστη απόκλιση μεταξύ των σημείων και της γραμμής. Η μη γραμμικότητα εκφράζεται συνήθως ως η απόκλιση του εύρους του αισθητήρα.

1.5.7 Ευαισθησία στη μέτρηση

Πρόκειται για το λόγο της μεταβολής στην ένδειξη του αισθητήρα, προς τη μεταβολή του φυσικού μεγέθους που την προκάλεσε. Από τον ορισμό αυτό προκύπτει ότι η ευαισθησία ισούται με την κλίση της γραφικής παράστασης του Σχήματος 1.6.

1.5.8 Ευαισθησία στη διαταραχή

Η βαθμονόμηση και τα χαρακτηριστικά ενός αισθητήρα ισχύουν όταν αυτό λειτουργεί εντός συγκεκριμένου εύρους περιβαλλοντικών συνθηκών παραμέτρων όπως η θερμοκρασία, η πίεση, η σχετική υγρασία κ.λπ. Το εύρος καθορίζεται από τον κατασκευαστή του αισθητήρα.



Σχήμα 1-7 Ολίσθηση του μηδενός, (b) Ολίσθηση ευαισθησίας, (c) Συνδυασμένη επίδραση των δύο ολισθήσεων

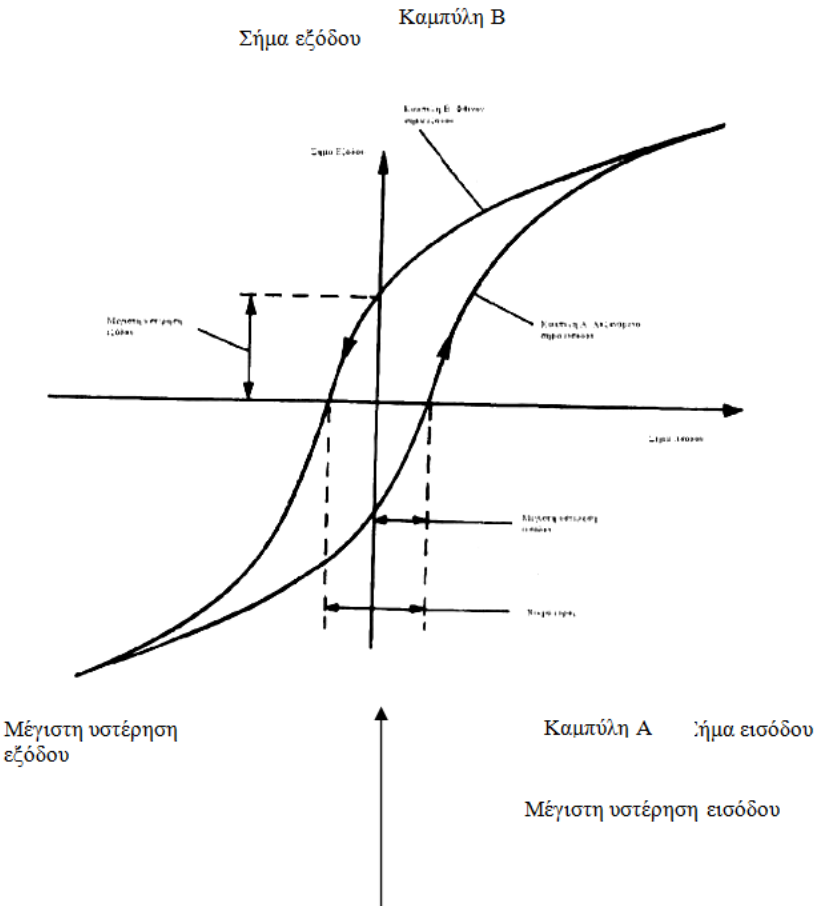
Μεταβολή κάποιας από τις παραμέτρους αυτές ενδέχεται να μεταβάλλει κάποιο από τα στατικά χαρακτηριστικά του αισθητήρα. Η μεταβολή αυτή ορίζεται ως η ευαισθησία στη

διαταραχή. Τα χαρακτηριστικά του αισθητήρα που μεταβάλλονται είναι κυρίως δύο και είναι γνωστά ως ολίσθηση του μηδενός (zero drift) και ολίσθηση ευαισθησίας (sensitivity drift).

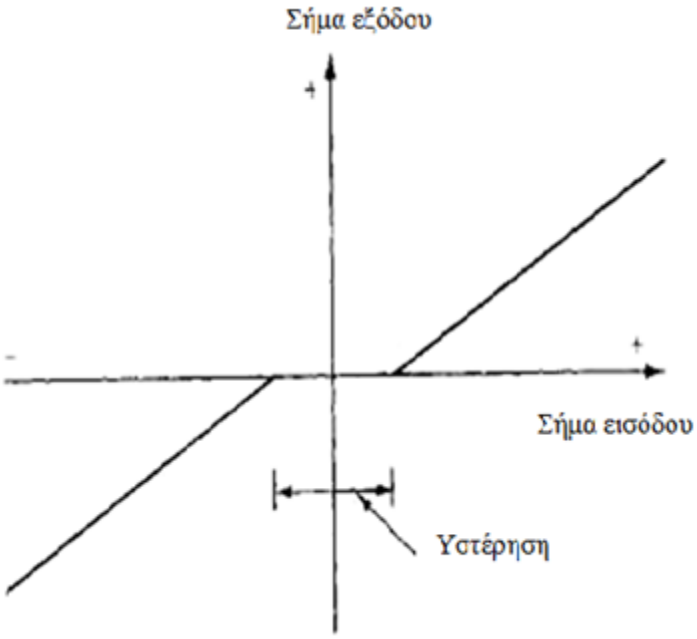
Η ολίσθηση του μηδενός είναι το μη μηδενικό σήμα εξόδου του αισθητήρα όταν το σήμα εισόδου είναι μηδέν, λόγω μεταβολής των περιβαλλοντικών συνθηκών. Μετριέται συνήθως σε $°C^{-1}$ στην περίπτωση π.χ. βολτομέτρου το οποίο έχει επηρεαστεί από τη μεταβολή της θερμοκρασίας. Αν ένας αισθητήρας επηρεάζεται από περισσότερες της μιας περιβαλλοντικές παραμέτρους, τότε αυτός χαρακτηρίζεται από αντίστοιχες σε αριθμό ολισθήσεις του μηδενός, μία για κάθε που επηρεάζει τον αισθητήρα παραμέτρου. Το Σχήμα 1.7 παρουσιάζει τη χαρακτηριστική ολίσθηση του μηδενός σε αισθητήρα πίεσης. Η ολίσθηση ευαισθησίας ορίζεται ως το ποσό μεταβολής της ευαισθησίας ενός αισθητήρα λόγω μεταβολής των περιβαλλοντικών συνθηκών. Εκφράζεται μέσω συντελεστών ολίσθησης ευαισθησίας, οι οποίοι εκφράζουν το μέγεθος της ολίσθησης ανά μονάδα μεταβολής της περιβαλλοντικής παραμέτρου που την προκάλεσε. Το Σχήμα 1.7b δείχνει την επίδραση της η ολίσθησης ευαισθησίας στα χαρακτηριστικά εξόδου ενός αισθητήρα. Η συνδυασμένη επίδραση των ολισθήσεων μηδενός και ευαισθησίας στα χαρακτηριστικά εξόδου του αισθητήρα απεικονίζεται στο Σχήμα 1.7c.

1.5.9 Υστέρηση

Στο Σχήμα 1.8 φαίνεται το σήμα εξόδου ενός αισθητήρα ο οποίος παρουσιάζει υστέρηση. Αν η τιμή του σήματος εισόδου μεταβάλλεται σταθερά, ξεκινώντας από αρνητικές τιμές, το σήμα εξόδου περιγράφεται από την καμπύλη Α. Αν κατόπιν το σήμα εξόδου μειώνεται σταδιακά, τότε το σήμα εξόδου περιγράφεται από την καμπύλη Β. Η μη ταύτιση των δύο καμπύλων «φορτίσεως - εκφορτίσεως» είναι γνωστή ως υστέρηση. Η υστέρηση εκφράζεται μέσω της μέγιστης υστέρησης εισόδου και της μέγιστης υστέρησης εξόδου, οι οποίες ορίζονται όπως φαίνεται στο Σχήμα 1.8.



Σχήμα 1-8 Υστέρηση Εξόδου



Σχήμα 1-9 Χαρακτηριστικά εξόδου αισθητήρα με νεκρό εύρος

1.5.10 Νεκρό εύρος

Ως νεκρό εύρος (dead space) ορίζεται το εύρος του σήματος εισόδου, για το οποίο το σήμα εξόδου είναι μηδενικό, όπως φαίνεται και στο Σχήμα 1.9. Όπως βλέπουμε και στο σχήμα, κάθε αισθητήρας ο οποίος εμφανίζει υστέρηση εμφανίζει και νεκρό εύρος. Παρ' όλα αυτά ακόμη και αισθητήρες οι οποίοι δεν εμφανίζουν σημαντική υστέρηση, μπορεί να εμφανίζουν νεκρό εύρος.

1.5.11 Κατώφλι

Αν το σήμα εισόδου ενός αισθητήρα αυξάνεται σταδιακά ξεκινώντας από μηδενική τιμή, αυτό "θα πρέπει να λάβει μία ορισμένη - μη μηδενική- τιμή πριν ο αισθητήρας δώσει κάποιο μη μηδενικό σήμα εξόδου. Αυτή η ελάχιστη τιμή του σήματος εισόδου, ονομάζεται κατώφλι του αισθητήρα. Το κατώφλι άλλοτε δίνεται ως απόλυτη τιμή και άλλοτε ως ποσοστό του εύρους εξόδου του αισθητήρα.

1.5.12 Διακριτική ικανότητα

Ως διακριτική ικανότητα ενός αισθητήρα ορίζεται η απαιτούμενη ελάχιστη μεταβολή του σήματος εισόδου, ώστε να προκληθεί μεταβολή στο σήμα εξόδου του αισθητήρα.

1.6 Αισθητήρες θερμοκρασίας

Το αισθητήριο θερμοκρασίας χρησιμοποιείται σε ένα σύστημα για να μετατρέψει τη θερμοκρασία του συστήματος σε ηλεκτρική τάση. Βασίζεται στη μεταβολή της τάσης στους ακροδέκτες μιας επαφής PN σε συνάρτηση με τη θερμοκρασία.

Για τη μέτρηση της θερμοκρασίας ενός συστήματος απαιτείται ιδιαίτερη προσοχή στην επιλογή του κατάλληλου αισθητηρίου έτσι ώστε να διασφαλίζεται η αξιοπιστία του ελέγχου και τη σωστή λειτουργία του συστήματος αυτού.

Τα πιο συνηθισμένα αισθητήρια μέτρησης της θερμοκρασίας είναι:

- Τα θερμοζεύγη
- Οι αντιστάσεις ανίχνευσης θερμοκρασίας (RTDs)
- Τα θερμίστορ θετικού και αρνητικού συντελεστή θερμοκρασίας.
- Τα πυρόμετρα
- Τα διμεταλλικά στοιχεία.
- Τα αισθητήρια διαστολής υγρών και αερίων
- Τα ημιαγωγικά αισθητήρια θερμοκρασίας κ.α

Από τα αισθητήρια θερμοκρασίας που χρησιμοποιούνται περισσότερο είναι οι πρώτες τέσσερις παραπάνω κατηγορίες και κυρίως τα θερμοζεύγη και οι αντιστάσεις ανίχνευσης θερμοκρασίας (RTDs), τα οποία χρησιμοποιούνται ευρύτατα στη βιομηχανία λόγω των ιδιαίτερων χαρακτηριστικών τους. Για τα αισθητήρια αυτά στους σύγχρονους ψηφιακούς ελεγκτές θερμοκρασίας υπάρχουν τυποποιημένες υποδοχές

Σχήμα 1-10 α) Αντιστάσεις ανίχνευσης θερμοκρασίας (RTDs) β) Θερμίστορ

1.6.1 Θερμίστορ (thermistor)

Τα θερμίστορ είναι στερεοί αισθητήρες θερμοκρασίας που συμπεριφέρονται όπως οι θερμικές αντιστάσεις. Το όνομα τους προέρχεται από τη σύνθεση των λέξεων θερμικό και αντιστάτης. Τα θερμίστορ είναι πολύ ευαίσθητα στη μεταβολή της θερμοκρασίας. Χρησιμοποιούνται στις συσκευές ως αισθητήρες αντίληψης, ελέγχου και διορθώσεως της θερμοκρασίας.

Σχήμα 1-11 Διάφορα σχήματα θερμίστορ

Οι αντιστάσεις θερμίστορ είναι βασικά δύο τύπων, οι NTC (Negative Temperature Coefficient) οι οποίες μικραίνουν την αντίσταση τους με την αύξηση της θερμοκρασίας και οι PTC (Positive

Temperature Coefficient) οι οποίες αυξάνουν την αντίσταση τους με την αύξηση της θερμοκρασίας.

NTC

Κατασκευάζονται από οξείδια των στοιχείων της ομάδας του σιδήρου όπως είναι τα οξείδια του χρωμίου (Cr), μαγγανίου (Mn), σιδήρου (Fe), χαλκού (Cu) ή νικελίου (Ni). Τα οξείδια αυτά έχουν μεγάλη ειδική αντίσταση και μπορεί κάτω από ορισμένες συνθήκες να μετατραπούν σε ημιαγωγούς τύπου P και N.

Τα οξείδια ανακατεύονται με συγκρατητική ύλη, μπαίνουν με πίεση σε ειδικά καλούπια, ψήνονται σε ειδικούς φούρνους όπου σε υψηλή θερμοκρασία γίνεται σύντηξη του οξειδίου, τοποθετούνται οι ακροδέκτες, επιστρώνονται με μονωτικό υλικό, μπαίνουν τα χρώματα που δείχνουν τα χαρακτηριστικά τους και δίνονται στο εμπόριο.

Τα θερμίστορ για τη μέτρηση θερμοκρασιών έχουν τα εξής χαρακτηριστικά:

Τα θερμίστορ σε σχήμα χάντρας είναι κατάλληλα για την ανίχνευση θερμοκρασιών σε πολύ στενά επίπεδα ή για επιφάνειες ή για εσωτερικές θερμοκρασίες.
Η θερμοκρασία των θερμίστορ είναι πολύ μικρή ώστε να είναι πολύ ευαίσθητα σ' ελάχιστες ή ακαριαίες μεταβολές θερμοκρασίας και υπάρχουν για όλες σχεδόν τις χρονικές καθυστερήσεις.
Είναι δυνατόν να μετρηθούν θερμοκρασίες υλικών χωρίς να υπάρχει επαφή και επίσης να μετρηθούν θερμοκρασίες από απόσταση ή να γίνει έλεγχος αυτών από απόσταση.

Δεν χρειάζονται αντισταθμικές επαφές ούτε κρύες ενώσεις.
Όσο μεγαλύτερη είναι η ηλεκτρική αντίσταση του θερμίστορ τόσο μικρότερο είναι το σφάλμα που εισάγεται από τους μεγάλους ακροδέκτες οι οποίοι δεν μπορούν να ξεπερνούν τα 10 μέτρα.
Τα θερμίστορ κατασκευάζονται με σταθερή σύνθεση έτσι ώστε με την γήρανση τους να μην παρουσιάζουν αποκλίσεις και να δίνουν σταθερές ενδείξεις.
Για χρήσεις σε θερμοκρασίες κάτω των $150^0 C$ μπορεί να παρουσιάζουν ένα σφάλμα της τάξης των $\pm 1\%$.
Τα θερμίστορ μπορεί να χρησιμοποιηθούν για πλήθος μετρήσεων θερμοκρασίας σε όργανα όπως είναι τα θερμόμετρα, υγρόμετρα, όργανα ελέγχου θερμοκρασίας, ροόμετρα, ροόμετρα υγρών, μετρητές κενού, ανιχνευτές αερίων, συναγερμοί για φωτιά, ανιχνευτές πάγου κ.τ.λ.

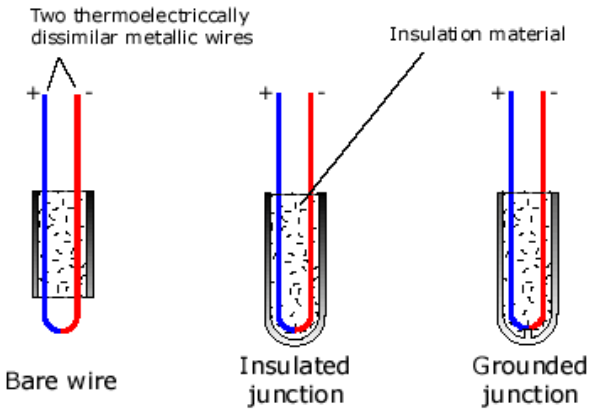
PTC

Τα θερμίστορ PTC (Positive Temperature Coefficient, θετικού συντελεστή θερμοκρασίας) χαρακτηρίζονται από τον υψηλό θετικό συντελεστή θερμοκρασίας. Αυτό βέβαια γίνεται μόνο για ορισμένες περιοχές θερμοκρασίας αφού για άλλες περιοχές ο συντελεστής είναι μηδέν ή αρνητικός. Τα υλικά κατασκευής των PTC είναι μείγματα ανθρακικού βαρίου ή οξείδια στροντίου και τιτανίου τα οποία ανακατεύονται μ' άλλα υλικά μαζί με συγκρατική ύλη. Το υλικό μπαίνει σε καλούπια, θερμαίνεται, προστίθενται οι ακροδέκτες και τα διάφορα χρώματα και δίνονται στο εμπόριο. Γενικά τα θερμίστορ χρησιμοποιούνται σαν εξαιρετικοί ρυθμιστές ρεύματος και θερμοκρασίας σε πολλές εφαρμογές.

1.6.2 Θερμοζεύγη (thermocouples)

Το θερμοζεύγος είναι ένας από τους πιο απλούς αισθητήρες. Είναι ένας ιδιαίτερα διαδεδομένος τύπος αισθητήρα τόσο στη βιομηχανία όσο και στην επιστήμη. Βασίζεται στην αλλαγή της αγωγιμότητας των υλικών του από τη θερμοκρασία. Αποτελείται από ένα ζεύγος ανόμοιων μετάλλων που συνδέονται μεταξύ τους στο σημείο της μετρήσεις. Μεταξύ των δύο καλωδίων παράγεται μια μικρή τάση.

Δύο θερμοηλεκτρικά ανόμοια μεταλλικά καλώδια



Σχήμα 1-12 Τύποι θερμοζευγών Ιδιότητες των θερμοζευγών

Οι ιδιότητες των θερμοζευγών είναι οι εξής:

- Ένα τρίτο μέταλλο δεν μπορεί να επηρεάσει κάποιο κύκλωμα θερμοηλεκτρικών ζευγών και να ασκήσει πάνω του κάποια επίδραση, υπό τον όρο ότι και οι δύο άκρες του θερμοζεύγους έχουν την ίδια θερμοκρασία. Αυτό σημαίνει ότι μπορεί να συγκολληθεί κάποιο άλλο μέταλλο στο θερμοηλεκτρικό ζεύγος χωρίς να το επηρεάσει, εφ' όσον δεν υπάρχει καμία καθαρή κλίση της θερμοκρασίας κατά μήκος του τρίτου μετάλλου.
- Το θερμοηλεκτρικό ζεύγος παράγεται από την μεταβολή της θερμοκρασίας κατά μήκος των καλωδίων και όχι στις συνδέσεις όπως θεωρείται συνήθως. Επομένως είναι σημαντική η ποιότητα του καλωδίου. Η ποιότητα των καλωδίων μπορεί να επηρεαστεί από τη μόλυνση στο λειτουργικό περιβάλλον και στο μονωτικό υλικό. Για τις θερμοκρασίες κάτω από τους 400°C, η μόλυνση των μονωμένων καλωδίων είναι σπάνιες. Στις θερμοκρασίες άνω των 1000°C, η επιλογή των υλικών μόνωσης και των θηκών, καθώς επίσης και το πάχος των καλωδίων, γίνονται κρίσιμες για τη σταθερότητα της βαθμολόγησης του θερμοηλεκτρικού ζεύγους
- Η τάση που παράγεται από ένα θερμοηλεκτρικό ζεύγος εξαρτάται από τη διαφορά θερμοκρασίας μεταξύ των συνδέσεων της μέτρησης και της αναφοράς.
- Απαιτείται η παραγωγή χαμηλής τάσης (50 mV) από τους αισθητήρες θερμοζευγών για να μην δημιουργηθεί πρόβλημα στις μηχανές, στους μετασχηματιστές και στα καλώδια.

- Η έκθεση των αισθητήρων θερμοζευγών σε υψηλή θερμοκρασία δημιουργεί οξείδωση και μείωση των ατμοσφαιρών που μπορεί να επηρρεάσει σημαντικά τα θερμοηλεκτρικά ζεύγη.

Πλεονεκτήματα και μειονεκτήματα των θερμοζευγών

Πλεονεκτήματα

- Λόγω των φυσικών χαρακτηριστικών τους, τα θερμοηλεκτρικά ζεύγη χρησιμοποιούνται σε πολλές εφαρμογές σε μεθόδους για τη μέτρηση της θερμοκρασίας.
- Δεν επηρεάζονται από τον κραδασμό και τη δόνηση, είναι απλά κατασκευασμένοι, δεν απαιτούν καμία δύναμη διέγερσης. Κανένας άλλος αισθητήρας θερμοκρασίας δεν παρέχει αυτόν τον βαθμό μεταβλητότητας.
- Τα θερμοηλεκτρικά ζεύγη είναι αισθητήρες που μπορούμε να πειραματιστούμε λόγω της ευρωστίας, του μεγάλου εύρους θερμοκρασίας που μπορούν να χρησιμοποιούν και των μοναδικών ιδιοτήτων τους.

Μειονεκτήματα

- Οι αισθητήρες θερμοζευγών παράγουν ένα σχετικά χαμηλό σήμα που είναι μη γραμμικό. Αυτό το χαρακτηριστικό απαιτεί μια ευαίσθητη και σταθερή συσκευή. Επίσης το χαμηλό επίπεδο σήματος απαιτεί να ληφθεί ένα πιο υψηλό επίπεδο προσοχής κατά την εγκατάσταση για να ελαχιστοποιήσουμε τις πιθανές πηγές θορύβου.
- Το μετρούμενο υλικό απαιτεί καλή ικανότητα απόρριψης του θορύβου. Αν δε γίνει αυτό μπορεί να δημιουργηθεί πρόβλημα με τα μη απομονωμένα συστήματα.

Είδη θερμοζευγών

Χρησιμοποιούνται περίπου 15 τύποι θερμοζευγών. Οι οκτώ από αυτούς είναι διεθνώς αναγνωρισμένοι. Στον παρακάτω πίνακα μπορούμε να δούμε τους τύπους αυτούς. Μερικά από τα μη αναγνωρισμένα θερμοζεύγη μπορούν να υπερέχουν σε ιδιαίτερες εφαρμογές θέσεων και έχουν κερδίσει έναν μεγάλο βαθμό αποδοχής και έχουν γίνει μερικώς αποδεκτοί από τη βιομηχανία. Κάθε τύπος θερμοζευγών έχει τα χαρακτηριστικά που μπορούν να χρησιμοποιηθούν σε διάφορες εφαρμογές. Η βιομηχανία προτιμά γενικά τους τύπους Κ και Ν λόγω της καταλληλότητας τους στις υψηλές θερμοκρασίες, ενώ άλλοι προτιμούν συχνά τον τύπο Τ λόγω της ευαισθησίας, του χαμηλότερου κόστους και της ευκολίας χρήσης του. Ο πίνακας 1.2 παρουσιάζει μια σειρά τυποποιημένων ειδών αισθητήρων θερμοζευγών:

Πίνακας 1-2 Σειρά τυποποιημένων ειδών των αισθητήρων θερμοζευγών

Τύπος	Θετικό υλικό	Αρνητικό υλικό	Κατηγορία ακρίβειας	Εύρος θερμοκρασίας °C	Σχόλια
B	Pt, 30%Rh	Pt, 6%Rh	0.5% >800°C	50 μέχρι 1820 (1 to 100)	Καλός στις υψηλές θερμοκρασίες.
C**	W, 5%Re	W, 26%Re	1% >425°C	0 μέχρι 2315 (0 to 870)	Πολύ μεγάλη χρήση στις υψηλές θερμοκρασίες, εύθραυστος.
D**	W, 3%Re	W, 25%Re	1% >425°C	0 μέχρι 2315 (0 to 260)	Πολύ μεγάλη χρήση στις υψηλές θερμοκρασίες, εύθραυστος.
E	Ni, 10%Cr	Cu, 45%Ni	0.5% or 1.7°C	-270 μέχρι 1000 (0 to 200)	Μεγάλη χρήση στις χαμηλές και μεσαίες θερμοκρασίες
G**	W	W, 26%Re	1% >425°C	0 μέχρι 2315 (0 μέχρι 260)	Πολύ μεγάλη χρήση στις υψηλές θερμοκρασίες, εύθραυστος.
J	Fe	Cu, 45%Ni	0.75% ή 2.2°C	-210 μέχρι 1200 (0 μέχρι 200)	Υψηλή θερμοκρασία, επηρεάζει το περιβάλλον
K*	Ni, 10%Cr	Ni, 2%Al 2%Mn 1%Si	0.75% ή 2.2°C	-270 μέχρι 1372 (0 μέχρι 80)	Υψηλή θερμοκρασία, οξειδωτικό περιβάλλον
L**	Fe	Cu, 45%Ni	0.4% ή 1.5°C	0 μέχρι 900	Παρόμοιος με το τύπο J, ξεπερασμένος, δεν χρησιμοποιείται πια
M**	Ni	Ni, 18%Mo	0.75% ή 2.2°C	-50 μέχρι 1410	
N*	Ni, 14%Cr 1.5%Si	Ni, 4.5%Si 0.1%Mg	0.75% ή 2.2°C	-270 μέχρι 1300 (0 μέχρι 200)	Νέος τύπος, αντικαθιστά τον K
P**	Platinel II	Platinel II	1.0%	0 μέχρι 1395	Σταθερό αλλά ακριβό υποκατάστατο των K& N
R	Pt, 13%Rh	Pt	0.25% ή 1.5°C	-50 μέχρι 1768 (0 μέχρι 50)	Ακρίβεια, υψηλή θερμοκρασία
S	Pt, 10%Rh	Pt	0.25% ή 1.5°C	-50 μέχρι 1768 (0 μέχρι 50)	Ακρίβεια, υψηλή θερμοκρασία
T*	Cu	Cu, 45%Ni	0.75% ή 1.0°C	-270 μέχρι 400 (-60 μέχρι 100)	Χαμηλή θερμοκρασία, ανθεκτικός στην υγρασία.
U**	Cu	Cu, 45%Ni	0.4% ή 1.5°C	0 μέχρι 600	Παρόμοιος με το τύπο T, ξεπερασμένος, δεν χρησιμοποιείται πια

1.6.3 Αισθητήρες θερμικών αντιστάσεων (Resistance Temperature Detector (RTD))

Στους αισθητήρες θερμικών αντιστάσεων **(Resistance Temperature Detector** (**RTD**) ή resistance thermometer), η αντίσταση αυξάνεται με την αύξηση της θερμοκρασίας.

Σχήμα 1-13 Αισθητήρες θερμικών αντιστάσεων (RTDs)

Στο εμπόριο βρίσκουμε RTDs από 10Ω έως 25ΚΩ. Ποιο διαδεδομένες είναι οι αντιστάτες λευκόχρυσου των 100, 200 και 1000Ω και οι αντιστάτες χαλκού των 1000Ω. Γενικά όσο ποιο μεγάλη είναι η αντίσταση του RTD τόσο πιο μικρές είναι οι διακυμάνσεις αντίστασης /τάσης στα καλώδια και στα κυκλώματα μολύβδου. Τα κοινά μέταλλα που χρησιμοποιούνται σε RTDs περιλαμβάνουν λευκόχρυσο, χαλκό, νικέλιο, BalcoTM (Fe 70%, Ni 30%), και το βολφράμιο. Το εύρος θερμοκρασίας ανάλογα με το υλικό κατασκευής τους παρατίθενται στον Πίνακα 1.3.

Πλεονεκτήματα και μειονεκτήματα

Πλεονεκτήματα

- Σταθερός και με μεγάλη ακρίβεια
- Η γραμμικότητα του είναι καλύτερη από αυτήν στα θερμοζεύγη (thermocouples)
- Έχει υψηλότερη αναλογία σήματος προς θόρυβο.

Μειονεκτήματα

- Υψηλό κόστος
- Απαιτεί μια τρέχουσα πηγή
- Ο χρόνος απόκρισης δεν είναι αρκετά γρήγορος για κάποιες εφαρμογές

Πίνακας 1-3 Κοινά μέταλλα που χρησιμοποιούνται σε RTDs

Υλικό	Εύρος Θερμοκρασίας	Σχόλια
Λευκόχρυσος (Pt)	-260~1000 °C (-440~1800 °F)	< 550 °C (1022 °F) στις περισσότερες εφαρμογές

Χαλκός (Cu)	-200~260 °C (-330~500 °F)	
Νικέλιο (Ni)	-200~430 °C (-330~800 °F)	Η γραμμικότητα δεν είναι καλή στις περισσότερες εφαρμογές
Balco (70% Ni-30% Fe)	-100~230 °C (-150~450 °F)	Η γραμμικότητα δεν είναι καλή. Φτηνός στην κατασκευή του
Βολφράμιο (W)	-100~1200 °C (-150~2200 °F)	

1.7 Αισθητήρες υγρασίας

1.7.1 Εισαγωγή

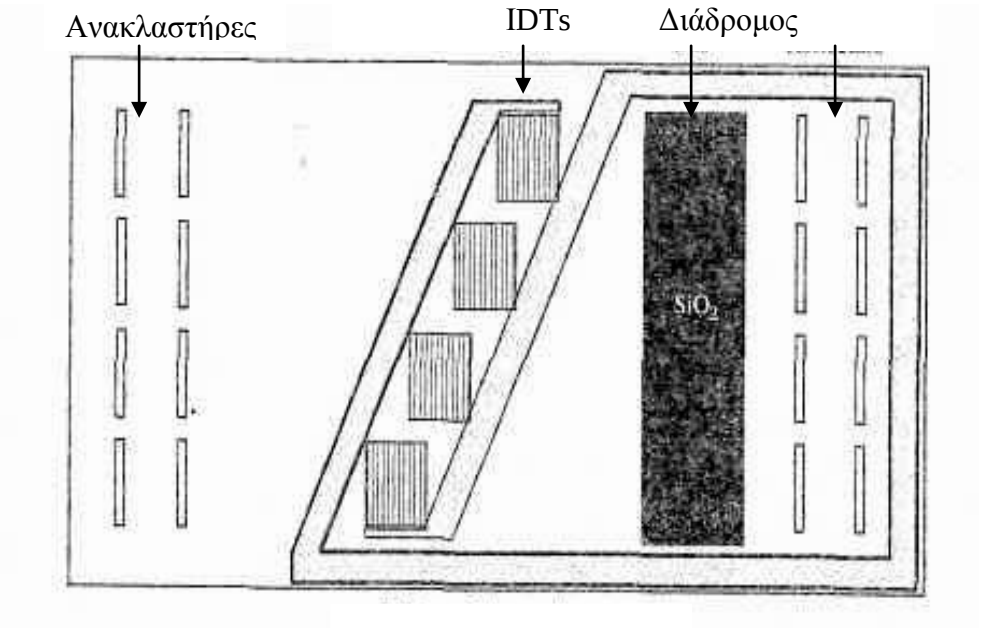
Η ανάπτυξη ενός μακρινού, ασύρματου, και παθητικού συστήματος αισθητήρων για τη μέτρηση υγρασίας είναι ακριβέστερη από τις συμβατικές μεθόδους. Εδώ, ένα τέτοιο σύστημα βασίζετε σε μια συσκευή SAW (Συσκευή επιφανειακών ακουστικών κυμάτων).

1.7.2 Λειτουργία

Τα τμήματα περιγράφουν ένα ασύρματο σύστημα αισθητήρων που μπορεί μέσο ενός παθητικού αισθητήρα SAW να μετρήσει τη σχετικής υγρασίας (RH) (Hollinger et Al 1999). Η γεννήτρια FM εκπέμπει συνεχώς σφυγμούς με διάρκεια 16,7ms που είναι γραμμικής συχνότητας και που διαμορφώνονται από 905 έως 925 MHz. Το αισθητήριο SAW επιτρέπει την ταυτόχρονη μέτρηση της θερμοκρασίας και της υγρασίας. Όπως φαίνεται στο Σχήμα 1.14, τέσσερα IDTs τοποθετούνται κατά μήκος του κέντρου του πιεζοηλεκτρικού υποστρώματος με τους ανακλαστήρες και στις δύο πλευρές.
Σε μια πλευρά, μεταξύ των IDTs και του συνόλου των ανακλαστήρων, τοποθετείται ένα ευαίσθητο υπόστρωμα (SiO_2 σε αυτήν την περίπτωση). Το υπόστρωμα μεταξύ των IDTs και του αριστερού συνόλου ανακλαστήρων αφήνεται γυμνό, δεδομένου ότι το νιοβικό (niobate) λίθιο παρουσιάζει πολύ μικρή μεταβολή στις αλλαγές της υγρασία.

Επομένως, η χωρίς επίστρωση πλευρά χρησιμοποιείται για τη μέτρηση της θερμοκρασίας, η οποία χρησιμοποιείται έπειτα για να αντισταθμίσει τις αλλαγές θερμοκρασίας στη μέτρηση υγρασίας από την ντυμένη πλευρά. Υπάρχουν μέχρι τρεις μοχλοί μετατόπισης φάσης (που δεν παρουσιάζονται) μπροστά από κάθε ανακλαστήρα.

Η μοναδική **ρύθμιση** αυτών των μοχλών μετατόπισης φάσης δίνει στους αισθητήρες μοναδικό αριθμό αναγνώρισης, ο οποίος μπορεί επίσης να καθοριστεί από το ασύρματο σύστημα βασισμένο στο απεικονισμένο ηλεκτρομαγνητικό σήμα. Ο κάθε διάδρομος (bus) συνδέεται με τα δύο τερματικά (terminals) κάθε IDT και αυτό συνδέεται επαγωγικά με την κεραία αισθητήρων **μέσω** κενού αέρα.



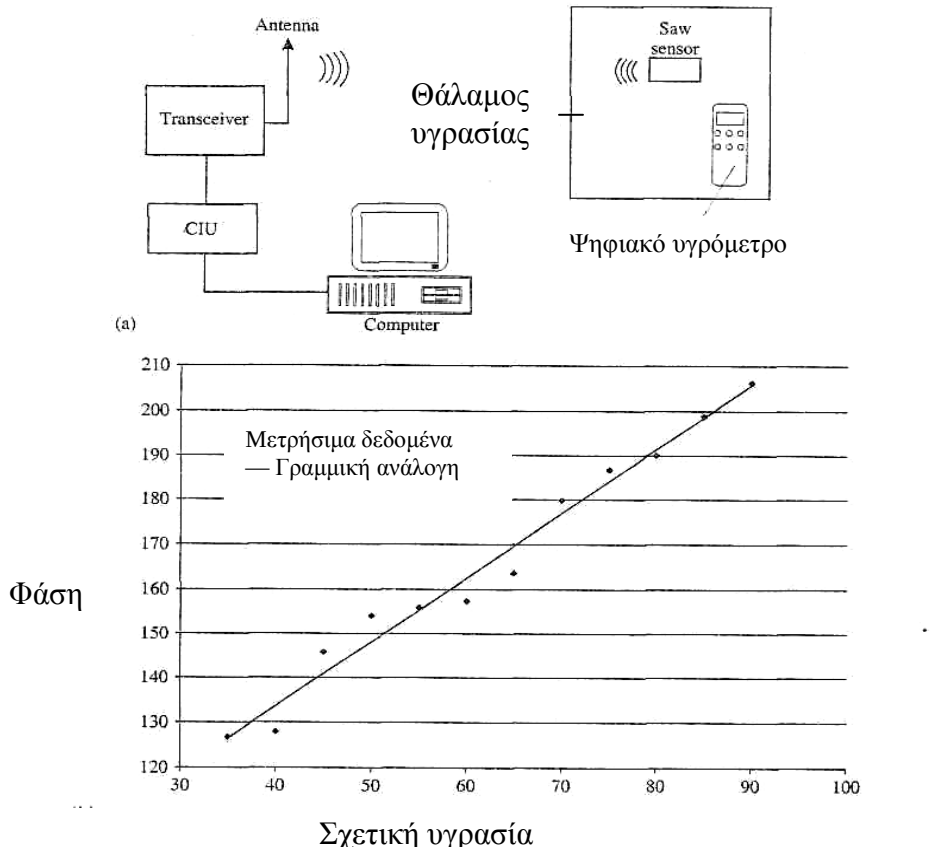
Σχήμα 1-14 Διάγραμμα των ασύρματων αισθητήρων SAW, για ταυτόχρονη μέτρηση θερμοκρασίας και υγρασίας για 915 MHz

Στο Σχήμα 1.15α παρουσιάζεται μια πειραματική οργάνωση για τη μακρινή μέτρηση υγρασίας χρησιμοποιώντας **τους ασύρματους** και παθητικούς αισθητήρες **SAW.** Τα κύρια συστατικά του συστήματος αποτελούνται από τον αισθητήρα SAW, τον πομποδέκτη, την κεντρική μονάδα διεπαφών (CIU), και τον υπολογιστή. Τοποθετείται επίσης ένας RH αισθητήρας (RH82) μέσα στην αίθουσα υγρασίας για τη βαθμολόγηση και την επαλήθευση.

Οι συσκευές SAW βασίζονται σε ένα υπόστρωμα $LiNbO_2$ με μέρος του υποστρώματος που ντύνεται με ένα λεπτό στρώμα διοξειδίου πυριτίου. Η γραμμή καθυστέρησης στον αισθητήρα SAW είναι τώρα ευαίσθητη στις αλλαγές της υγρασίας, δεδομένου ότι το διοξείδιο πυριτίου προσροφά την υγρασία από τον υγρό αέρα και ο πομποδέκτης εκπέμπει παλμούς RF, οι οποίοι παίρνονται από τον αισθητήρα SAW και μετατρέπονται σε ακουστικά κύματα στην επιφάνεια του αισθητήρα, τα οποία ανακλώνται περαιτέρω από τους μεμονωμένα χωρισμένους κατά διαστήματα ανακλαστήρες.

Το ανακλασμένο σήμα που παραλαμβάνεται από τον πομποδέκτη, περιέχει τις πληροφορίες των αισθητήρων. Αυτό το σήμα περνά μέσω του CIU, στο οποίο πραγματοποιείται όλη η επεξεργασία σήματος και έπειτα το επεξεργασμένο σήμα στέλνεται στον υπολογιστή μέσω της τμηματικής διεπαφής. Η διαφορά φάσης μεταξύ δύο ανακλαστήρων κατά μήκος της πορείας διάδοσης μπορεί

να χρησιμοποιηθεί ως μέτρο της υγρασίας. Η ακόλουθη γραφική παράσταση, Σχήμα 1.15β, παρουσιάζει την αλλαγή φάσης στον αισθητήρα υγρασίας, όπως μετριέται από το συγκρότημα πομποδεκτών και ηλεκτρονικών υπολογιστών. Διαπιστώνεται ότι η αλλαγή φάσης ποικίλλει γραμμικά.



Σχήμα 1-15 (α) Μέτρηση υγρασίας
(β) Η επίδραση της σχετικής υγρασίας στη φάση

1.7.3 Υλικά κατασκευής

Το υλικό υποστρωμάτων για τη συσκευή SAW αποτελείται από YZ-$LiNbO_3$, το οποίο είναι μια περικοπή Y-άξονα και Z-άξονα διάδοσης του λιθίου του κρυστάλλου. Το μέγεθος του πιεζοηλεκτρικού υποστρώματος είναι περίπου 4,3 χιλ. και πάχος 0,5 χιλ. Τα IDTs και οι ανακλαστήρες αποτελούνται από αργίλιο.

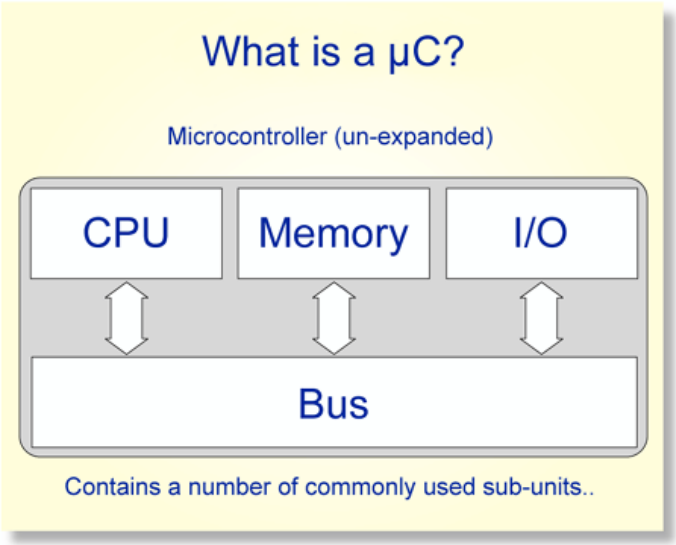
2. ΚΕΦΑΛΑΙΟ 2

ΜΙΚΡΟΕΛΕΓΚΤΕΣ

2.1 Εισαγωγή

Βασικά ένας μικροελεγκτής είναι μία συσκευή η οποία ενσωματώνει έναν αριθμό από εξαρτήματα ενός συστήματος μικροεπεξεργαστή, σε ένα και μόνο μικροτσίπ. Οπότε, ένας μικροελεγκτής συνδυάζει στο ίδιο μικροτσίπ:

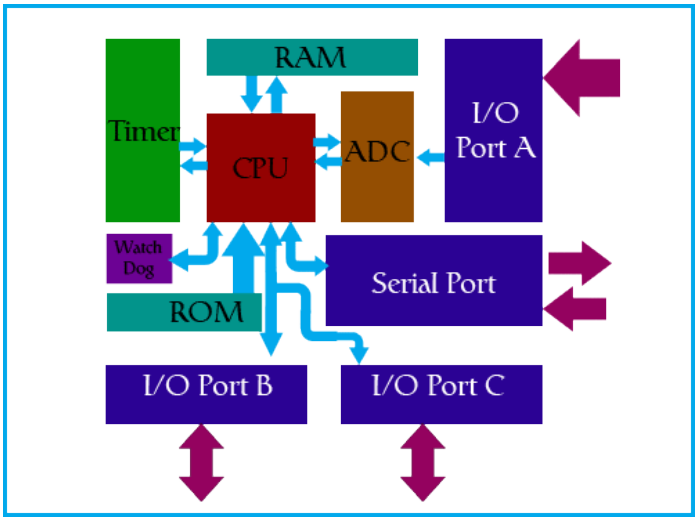
- Τη CPU (Κεντρική Μονάδα Επεξεργασίας)
- Μνήμη (RAM και ROM)
- Κάποιες παράλληλες ψηφιακές «πόρτες» I/O



Σχήμα 2-1 Κύρια Χαρακτηριστικά ενός μικροελεγκτή

Οι περισσότεροι μικροελεγκτές θα συνδυάζουν συσκευές όπως:

- Μονάδα Timer που θα επιτρέπει στο μικροελεγκτή να εκτελέσει κάποιες εργασίες για συγκεκριμένες χρονικές περιόδους.
- Κάποιες σειριακές «πόρτες» I/O για να επιτρέπουν την ροή δεδομένων μεταξύ του μικροελεγκτή και άλλων συσκευών όπως Η/Υ ή άλλων μικροελεγκτών.
- Ένα ADC (Analog to Digital Converter) μετατροπέα, για να επιτρέπει στο μικροελεγκτή να δέχεται δεδομένα σε αναλογική μορφή για επεξεργασία.

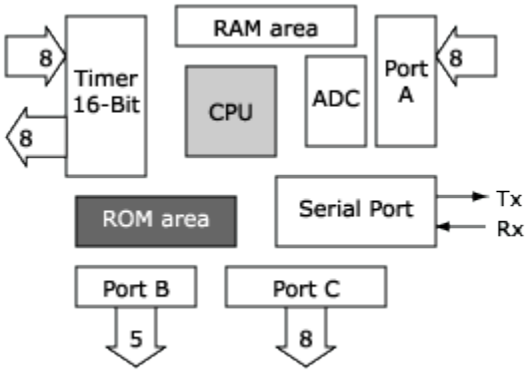


Σχήμα 2-2 Ένας μικροελεγκτής μονού chip.

Το Σχήμα 2.2 παρουσιάζει μία τυπική συσκευή μικροελεγκτή και τις διαφορετικές υπό-μονάδες ενσωματωμένες μέσα στο μικροτσίπ του μικροελεγκτή. Η καρδιά του μικροελεγκτή είναι η ο πυρήνας CPU.

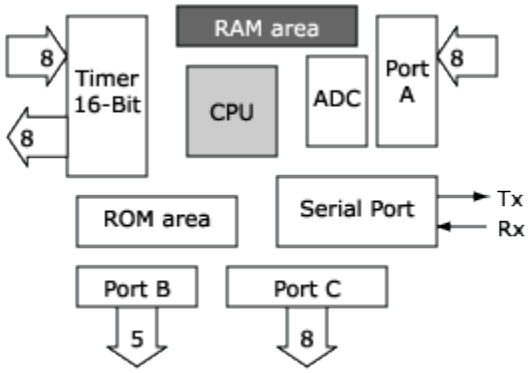
2.2 Η μνήμη στον μικροελεγκτή

Η ποσότητα της μνήμης που περιέχεται μέσα σε ένα μικροελεγκτή ποικίλει μεταξύ διαφορετικών μικροελεγκτών. Κάποιοι μπορεί να μην έχουν καθόλου ενσωματωμένη μνήμη(π.χ Hitachi 6503). Παρόλα αυτά, οι περισσότεροι μοντέρνοι μικροελεγκτές έχουν ενσωματωμένη μνήμη η οποία χωρίζεται σε ROM και RAM, με τυπικά περισσότερη ROM από RAM.



Σχήμα 2-3 Η ROM ενός μικροελεγκτή

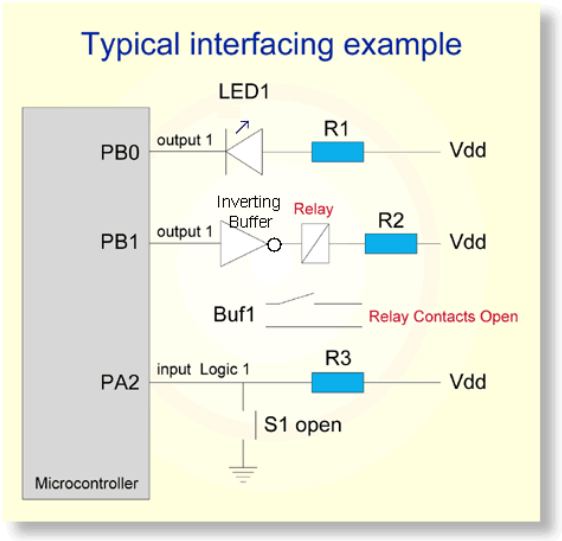
Η μνήμη τύπου ROM, χρησιμοποιείται για την αποθήκευση του κώδικα προγράμματος. Η μνήμη ROM μπορεί να είναι EPROM (one time programmable memory) ή EEPROM (electronically erasable programmable memory)
Η μνήμη τύπου RAM, χρησιμοποιείται για αποθήκευση πληροφοριών και για εργασίες διαχείρισης στιβάδας (stack management stack). Επίσης, χρησιμοποιείται για στιβάδες καταχωρητών (register stacks).



Σχήμα 2-4 Η RAM ενός μικροελεγκτή

2.3 Οι θύρες Επικοινωνίας I/O

Οι ψηφιακές θύρες επικοινωνίας, είναι τα μέσα μέσω των οποίων ο μικροελεγκτής αλληλεπιδρά με το περιβάλλον. Οι ψηφιακές θύρες συνηθίζεται να ομαδοποιούνται ανά πολλαπλάσια του byte οι οποίες μπορούν να ρυθμιστούν είτε ως είσοδοι, είτε ως έξοδοι. Μια τυπική διεπαφή θυρών μοιάζει κάπως έτσι:



Σχήμα 2-5 Τυπικό παράδειγμα διεπαφής

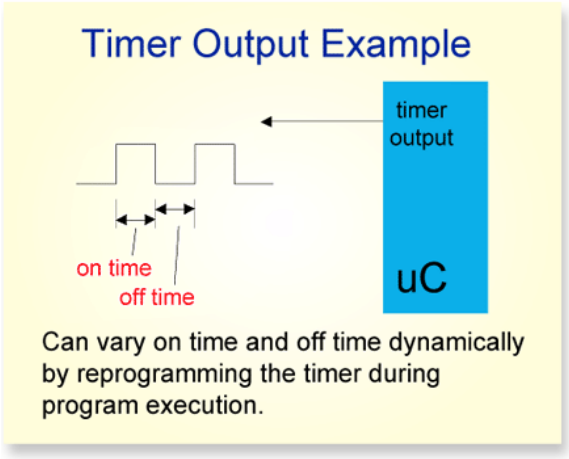
Οι θύρες του μικροελεγκτή, μπορούν να χρησιμοποιηθούν για το χειρισμό LED ή ρελέ, όπως επίσης για τον έλεγχο κατάστασης διακοπτών και τον έλεγχο λογικών κυκλωμάτων.

2.4 Περιφερειακές Μονάδες σε ένα Τυπικό Μικροελεγκτή

Οι περισσότεροι μικροελεγκτές, περιέχουν έναν αριθμό από μονάδες. Στο παρελθόν πολλές από αυτές σχεδιάζονταν σαν ξεχωριστά τσιπ σε ένα συμβατικό σύστημα μικροελεγκτή. Ενσωματώνοντας τα σε ένα και μόνο τσιπ, μας επιτρέπεται περισσότερη λειτουργικότητα με ένα μόνο τσιπ και καταλαμβάνοντας λιγότερο χώρο.
Τυπικές συσκευές είναι:

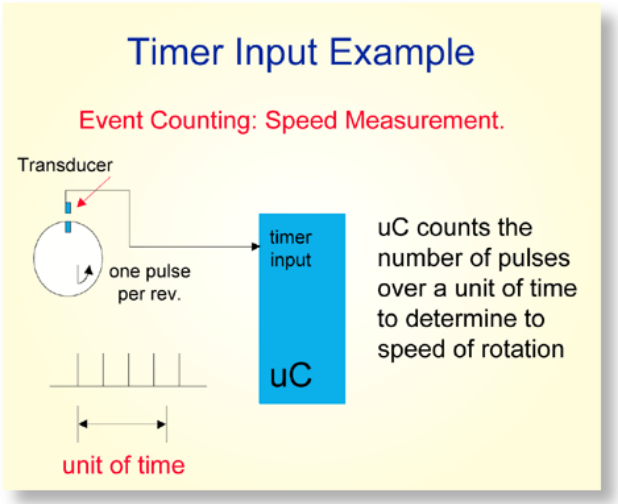
- Μονάδα Timer
- Σειριακή θύρα I/O
- ADC (Analog to Digital Converter)

2.4.1 *Μονάδες Timer*



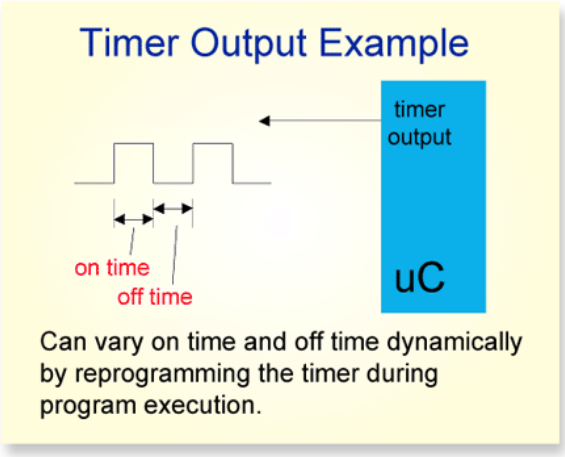
Σχήμα 2-6 Παράδειγμα Εξόδου του Timer

Μία κοινή ανάγκη ενός μικροελεγκτή, είναι να μπορεί να «ανάψει» μία συσκευή (πχ LED,RELAY) για μια χρονική περίοδο, και μετά από λίγο να την ξανά σβήσει. Αυτή η διαδικασία είναι επεξεργαστικά δαπανηρή, με την έννοια ότι ο επεξεργαστής θα μπορούσε να κάνει άλλα πράγματα την ώρα που περιμένει τον ενδεδειγμένο χρόνο να περάσει για να σβήσει τη συσκευή. Εναλλακτικά θα μπορούσε να χρησιμοποιηθεί μία μονάδα Timer που θα αναλάμβανε αυτή την εργασία, και έτσι ο επεξεργαστής θα μπορούσε να σπαταλήσει την ισχύ του πιο παραγωγικά. Οι περισσότεροι μικροελεγκτές έχουν τουλάχιστον μία μονάδα Timer με πολλαπλές εισόδους και εξόδους.
Οι είσοδοι επιτρέπουν στον Timer να υπολογίζει τον χρόνο ενός σήματος που δέχεται στην είσοδο του όπως δείχνει το παρακάτω παράδειγμα.



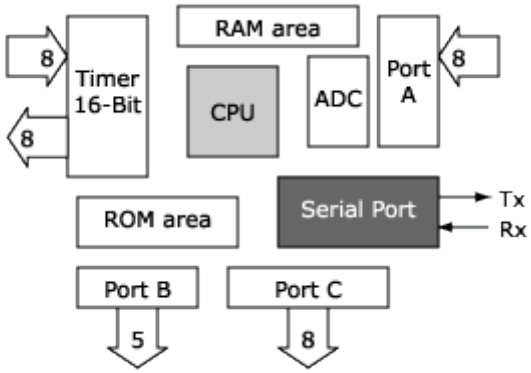
Σχήμα 2-7 Παράδειγμα Εισόδου στον Timer

Στο συγκεκριμένο παράδειγμα, ο περιστρεφόμενος άξονας παράγει έναν παλμό κάθε περιστροφή. Ο μικροελεγκτής μετρά τον χρόνο που χρειάζεται μία πλήρης περιστροφή και έτσι μπορεί και υπολογίζει τη γωνιακή ταχύτητα του άξονα.



Το παραπάνω σχέδιο παρουσιάζει πως ένας Timer μπορεί να παράγει έναν παλμό στην έξοδο του, πιθανό για να χειριστεί κάποια εξωτερική συσκευή.

2.4.2 Η Σειριακή Θύρα



Σχήμα 2-8 Η Σειριακή Θύρα

Κάποιοι μικροελεγκτές έχουν μία σειριακή θύρα η οποία τους επιτρέπει να στέλνουν δεδομένα σε κάποιον άλλο μικροελεγκτή, Η/Υ ή κάποιο άλλο απομακρυσμένο σύστημα μέσω ενός ζεύγους καλωδίων. Αυτός μπορεί να είναι ένας πολύ βολικός τρόπος αποστολής δεδομένων μεταξύ δύο συσκευών. Ένα μειονέκτημα όμως είναι η χαμηλή ταχύτητα μεταφοράς δεδομένων μέσω της σειριακής θύρας. Ο ρυθμός μεταφοράς μπορεί να προγραμματιστεί σε από 300 bits/second έως 1115200 bits/second.
Συνήθως, αναφερόμαστε στη σειριακή θύρα ως Serial Communications Interface(SCI). Οι περισσότερες SCI μονάδες που περιέχονται στα τσιπ του μικροελεγκτή, είναι υποσυστήματα μίας πιο παραδοσιακής μονάδας. Της Universal Asyncronous Reciever/Transmitter(UART) η

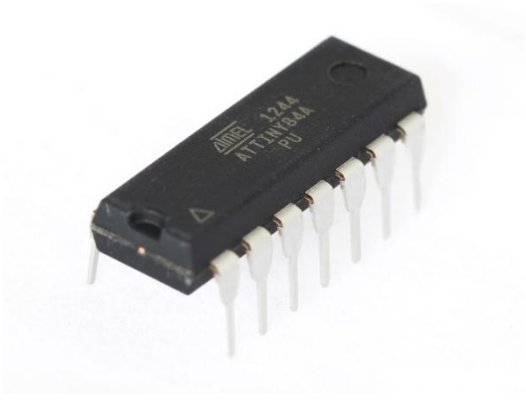
οποία μονάδα υπάρχει στους κλασικούς Η/Υ. Οι περισσότερες SCI μονάδες μπορούν να λειτουργήσουν σε ασύγχρονη λειτουργία, ενώ κάποιες υποστηρίζουν και σύγχρονη λειτουργία.

2.4.3 Analog to digital Converter (ADC)

Στον πραγματικό κόσμο, τα φυσικά «σήματα» είναι συχνά αναλογικά. Παραδείγματος χάρη, μπορεί να θέλουμε να παρακολουθήσουμε/καταγράψουμε τα σήματα από έναν μετρητή τάσης ή από κάποιο αισθητήριο που μετατρέπει τη θερμοκρασία σε ηλεκτρικό αναλογικό σήμα. Εφόσον λοιπόν πολλές φορές χρειάζεται ένας μικροελεγκτής για αυτού του είδους τις εργασίες, πολλοί έχουν ενσωματωμένο έναν ADC (μετατροπέα από αναλογικό σε ψηφιακό σήμα).
Ένας ADC είναι ασύνηθες να παράγει πάνω από 10 bits. Για πιο απαιτητικές εφαρμογές, η απόδοση του ADC χρειάζεται περεταίρω εξέταση.
Γενικότερα, οι περισσότεροι 8-μπιτοι (8-bit) μικροελεγκτές έχουν μειωμένη απόδοση, λόγω της μειωμένης ανάλυσης του ADC και της χαμηλής απόδοσης του επεξεργαστή (CPU). Είναι περιορισμένοι σε απλές καταγραφές δεδομένων (data logging) και εφαρμογές μέτρησης μικρής ακρίβειας. Οι περισσότεροι 8-μπιτοι (8-bit) μικροελεγκτές είναι αδυνατούν να εκτελέσουν πολύπλοκους υπολογισμούς λόγω τους περιορισμένου σετ εντολών που διαθέτουν και της χαμηλής ταχύτητας τους.

2.5 Διαδεδομένες Κατηγορίες Μικροελεγκτών

- Μικροελεγκτές (καμιά φορά 4-bit αλλά συνήθως 8-bit) πολύ χαμηλού κόστους, γενικής χρήσης, με πολύ μικρό αριθμό ακροδεκτών (ακόμη και λιγότερους από 8). Σχεδιάζονται με έμφαση στη χαμηλή κατανάλωση ισχύος και την αυτάρκεια, ώστε να χρειάζονται ελάχιστα ή και καθόλου εξωτερικά εξαρτήματα και να μη μπορεί να αντιγραφεί εύκολα το εσωτερικό λογισμικό τους. Απουσιάζει η δυνατότητα επέκτασης της μνήμης τους. Μερικά μοντέλα είναι ευρέως γνωστά στους ερασιτέχνες ηλεκτρονικούς, όπως πχ οι περισσότεροι μικροελεγκτές των σειρών PIC (Microchip), AVR (Atmel) και 8051 (Intel, Atmel, Dallas κα)

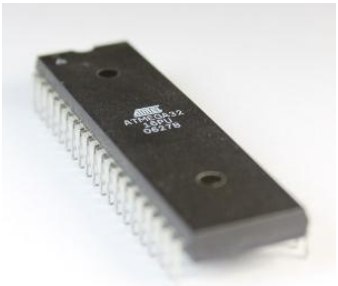


Σχήμα 2-9 Atmel 1244 8-bit AVR microcontroller

- Μικροελεγκτές (συνήθως 8-bit αλλά και 16 ή 32-bit) χαμηλού κόστους, γενικής χρήσης, με μέτριο έως σχετικά μεγάλο αριθμό ακροδεκτών. Διαθέτουν μεγάλο αριθμό κοινών περιφερειακών, όπως θύρες UART, I2C, SPI ή CAN, μετατροπείς αναλογικού σε ψηφιακό και ψηφιακού σε αναλογικό. Στους κατασκευαστές της Άπω Ανατολής (Ιαπωνία, Κορέα), συνηθίζεται η ενσωμάτωση ελεγκτών οθόνης υγρών κρυστάλλων και πληκτρολογίου. Μερικές φορές παρέχουν δυνατότητα εξωτερικής επέκτασης της μνήμης τους.

Σχήμα 2-1016-bit NEC microcontroller

- Μικροελεγκτές (κυρίως 32-bit) μέσου κόστους, γενικής χρήσης, με μεγάλο αριθμό ακροδεκτών. Χαρακτηρίζονται από έμφαση στην ταχύτητα εκτέλεσης εντολών, υψηλή αυτάρκεια περιφερειακών και μεγάλες δυνατότητες εσωτερικής ή εξωτερικής μνήμης προγράμματος (FLASH) και RAM. Στο χώρο αυτό έχουν ισχυρή παρουσία οι αρχιτεκτονικές με υψηλή μεταφερσιμότητα λογισμικού (portability) από τον ένα στον άλλο κατασκευαστή. Πχ μεταξύ των μικροελεγκτών τύπου ARM ή MIPS, το σύνολο των βασικών εντολών που αναγνωρίζει η ALU είναι ακριβώς το ίδιο, μειώνοντας έτσι τις μεγάλες αλλαγές στο λογισμικό, όταν στο μέλλον ο πελάτης υιοθετήσει ένα μικροελεγκτή άλλου κατασκευαστή (αρκεί, φυσικά, να υποστηρίζει κι αυτός το σύνολο εντολών ARM ή MIPS, αντίστοιχα).



Σχήμα 2-1132-bit AVR microcontroller

- Μικροελεγκτές εξειδικευμένων εφαρμογών, οι οποίοι ενσωματώνουν συνήθως κάποιο εξειδικευμένο πρωτόκολλο επικοινωνίας το οποίο υλοποιείται πάντοτε σε hardware.

Τέτοιοι μικροελεγκτές χρησιμοποιούνται σε τηλεπικοινωνιακές συσκευές όπως τα μόντεμ. Η μεγάλη μερίδα πωλήσεων των μικροελεγκτών εξακολουθεί να αφορά αυτούς των 8-bit, καθώς είναι η κατηγορία με το χαμηλότερο κόστος και το μικρότερο μέγεθος λογισμικού για το ίδιο αποτέλεσμα, ιδίως επειδή οι σύγχρονες οικογένειες μικροελεγκτών 8-bit έχουν πολύ βελτιωμένες επιδόσεις σε σχέση με το παρελθόν.

2.6 Κατασκευαστές

Μερικοί από τους γνωστότερους κατασκευαστές μικροελεγκτών είναι οι:

- ARM (δεν κατασκευάζει αλλά παραχωρεί δικαιώματα χρήσης του πυρήνα)
- Atmel
- Epson
- Freescale Semiconductor (πρώην Motorola)
- Hitachi
- Maxim (μετά την εξαγορά της Dallas)
- Microchip
- NEC
- Toshiba
- Texas Instruments

2.7 Δημοφιλή Μοντέλα Μικροελεγκτών

Σήμερα, υπάρχουν πολλά διαφορετικά συστήματα μικροελεγκτών. Μερικά από τα πιο διαδεδομένα είναι τα παρακάτω:

2.7.1 Atmel AVR Development Boards

Είναι μία οικογένεια αναπτυξιακών πλακετών, με εύρος κατηγοριών ανάλογα τον μικροελεγκτή που περιέχει το κάθε σύστημα

Σχήμα 2-12 Atmel AVR development Board

Στην παραπάνω εικόνα απεικονίζεται η αναπτυξιακή πλακέτα με μικροελεγκτή ATMEL16, ένα 16-bit μικροελεγκτή. Η πλακέτα έχει 40 θήρες I/O για την αλληλεπίδραση με εξωτερικές μονάδες. Φυσικά, έχει ενσωματωμένο UART και USB θύρα για την επικοινωνία με τον Η/Υ.

2.7.2 Raspberry Pi

Σχήμα 2-13 Raspberry Pi B/B+

Το Raspberry Pi, είναι ένα πολύ διαδεδομένο ενοποιημένο σύστημα μικροελεγκτή, της οικογένειας BROADCOM, το οποίο περιέχει όλα τα βασικά στοιχεία τα οποία προαναφέραμε, όμως ενσωματώνει και πολλά άλλα περιφερειακά, όπως δεύτερη USB θύρα, LAN θύρα δικτύου, έξοδο video και άλλα. Αυτό καθιστά το συγκεκριμένο σύστημα κατάλληλο για περίπλοκες εφαρμογές και υλοποιήσεις. Παρόλα ταύτα όμως, το καθιστά αρκετά πιο ακριβό από αλλά συστήματα της ίδιας συνομοταξίας.

2.7.3 Arduino Family

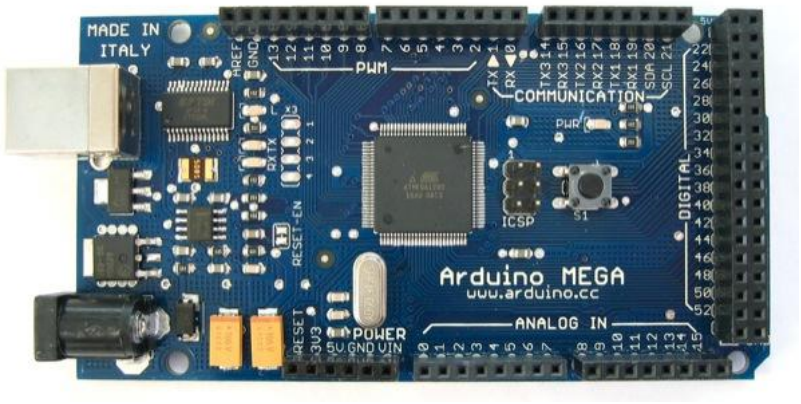
Η συγκεκριμένη οικογένεια αναπτυξιακών συστημάτων, είναι η πιο ευρέως διαδεδομένη. Είναι μία οικογένεια αναπτυξιακών πλακετών, φυσικά χωρισμένη ανά κατηγορίες, ανάλογα τον μικροελεγκτή που διαθέτει η κάθε πλακέτα. Δεν ενσωματώνει κανένα περιφερειακό εκτός από την θύρα USB, με την οποία επικοινωνεί με τον Η/Υ. Περεταίρω, ο χρήστης μπορεί να ενσωματώσει οποιαδήποτε περιφερειακή μονάδα επιθυμεί μέσω των ψηφιακών ή αναλογικών θηρών που διαθέτει το σύστημα, κατά το δοκούν.
Τα πιο διαδεδομένα συστήματα αυτής της οικογένειας είναι τα:

- **Arduino UNO**

Σχήμα 2-14 Arduino UNO

Το Arduino Uno είναι ένα ενσωματωμένο σύστημα μικροελεγκτή με βάση τον 16-bit ATmega328. Διαθέτει 14 ψηφιακές ακίδες εισόδου / εξόδου (εκ των οποίων 6 είναι δυνατόν να χρησιμοποιηθούν ως έξοδοι PWM), 6 αναλογικές εισόδους, ένα 16 MHz κεραμικό συντονιστή, μια σύνδεση USB, υποδοχή τροφοδοσίας, μια κεφαλίδα ICSP, και ένα κουμπί reset. Είναι ένα σύστημα που συνίσταται για εφαρμογές μικρής κλίμακας και χαμηλών απαιτήσεων λόγω των λίγων θηρών, και της χαμηλής μνήμης που διαθέτει. Παρόλα αυτά, είναι πολύ οικονομικό σε κόστος, όπως και οι περισσότερες περιφερειακές μονάδες του.

- **Arduino Mega**



Σχήμα 2-15 Arduino Mega

Το Arduino Mega είναι μια πλακέτα μικροελεγκτή με βάση τον 16-bit ATmega1280. Διαθέτει 54 ψηφιακές ακίδες εισόδου / εξόδου (εκ των οποίων 14 μπορούν να χρησιμοποιηθούν ως έξοδοι PWM), 16 αναλογικές εισόδους, 4 UARTs (σειριακές θύρες hardware), ένα 16 MHz ταλαντωτή κρυστάλλου, μια σύνδεση USB, υποδοχή ρεύματος, μια επικεφαλίδα ICSP, και ένα κουμπί reset. Αυτό το σύστημα συνίσταται για μεγαλύτερες και πιο απαιτητικές εφαρμογές, αφού έχει τετραπλάσια μνήμη από το προηγούμενο.

Όλα αυτά τα συστήματα, σε συνδυασμό με εξειδικευμένες μονάδες (όπως πχ. Αισθητήρια, ρελέ, LED, κινητήρες και άλλα) και τον ανάλογο προγραμματισμό από τον χρήστη, μπορούν εν δυνάμει να χρησιμοποιηθούν σε έναν πάρα πολύ μεγάλο αριθμό εφαρμογών και υλοποιήσεων. Οι μόνοι παράγοντες που μπαίνουν εμπόδιο στην υλοποίηση και την εξέλιξη αυτού, είναι αρχικά η ύπαρξη των αναγκαίων περιφερειακών μονάδων, η συμβατότητα με το εκάστοτε σύστημα που χρησιμοποιείται και φυσικά η ελευθερία στην πρόσβαση «πηγαίου κώδικα», «βιβλιοθηκών», λύσεων και γενικότερα ιδεών. Πράγμα το οποίο σπανίζει, λόγω του γενικού ανταγωνισμού, θεμιτού ή αθέμιτου.

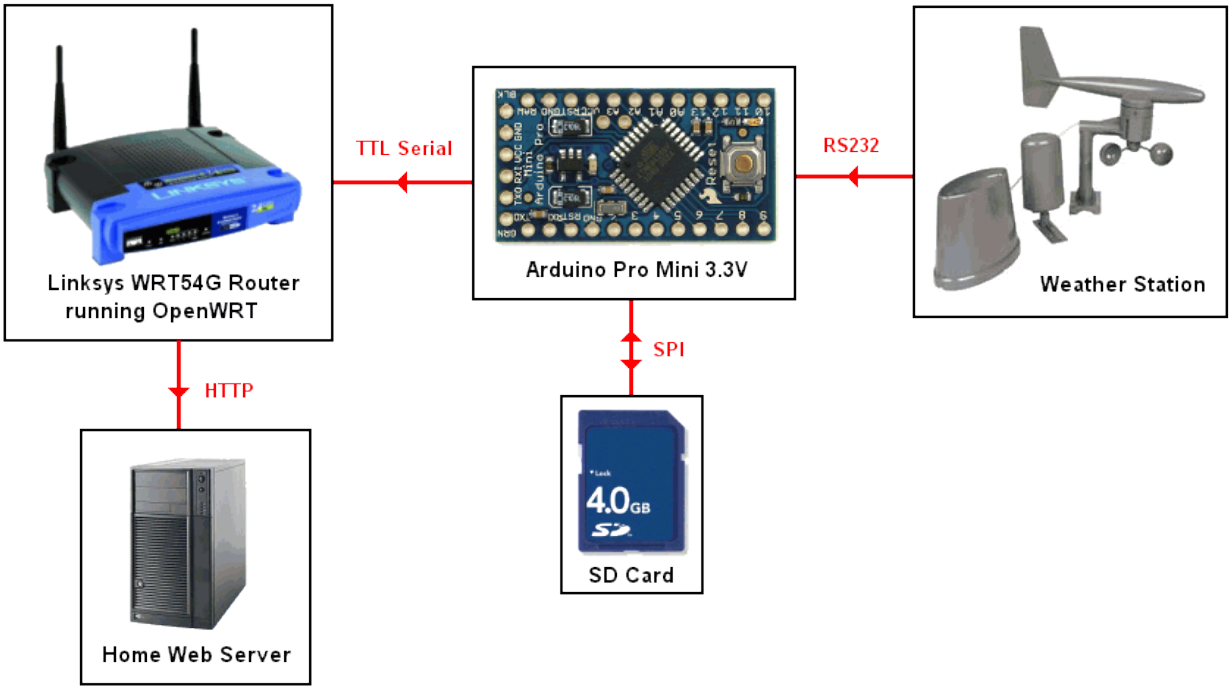
2.8 Πεδία Εφαρμογών

Στον πραγματικό κόσμο, η γενικότερη ανάγκη που εγείρεται στη βιομηχανία αλλά και στην καθημερινότητα, είναι η αυτοματοποίηση κάποιων εργασιών, όπου να καθιστά την ανάγκη για ανθρώπινη παρέμβαση στην όλη διαδικασία, όσο το δυνατό μικρότερη. Φυσικά στην κάθε υλοποίηση παίζει ρόλο και το κόστος κατασκευής. Για αυτό το λόγο, για υλοποιήσεις μικρού έως μεσαίου μεγέθους και απαιτήσεων, καταλληλότερα συστήματα είναι τα ενοποιημένα συστήματα μικροελεγκτών που προαναφέραμε. Μεγάλο ρόλο παίζει και η ευελιξία των συστημάτων αυτών εφόσον μπορούν να ενσωματώσουν ένα πολύ μεγάλο αριθμό από περιφερειακές μονάδες ανάλογα την περίσταση.
Παραδείγματα εφαρμογών τέτοιων συστημάτων παρατίθενται ακολούθως.

2.8.1 Σύστημα Παρακολούθησης Καιρικών Συνθηκών

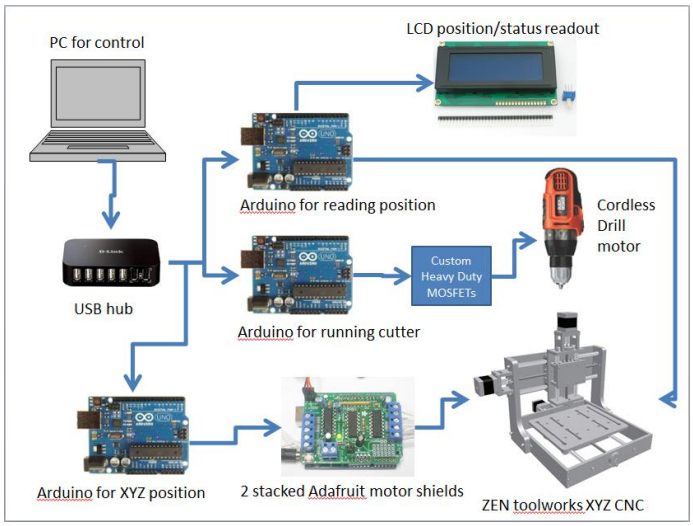
Ένα σύστημα με αισθητήρες θερμοκρασίας υγρασίας και βαρομετρικού, το οποίο μετράει καταγράφει και πιθανό να αποστέλλει τα δεδομένα σε κάποιον απομακρυσμένο

διακομιστή(Server).



Σχήμα 2-16 Arduino Pro Mini based Weather Station

2.8.2 Τρισδιάστατοι Εκτυπωτές/Ρούτερ CNC



Σχήμα 2-17 3D CNC printer

Τον τελευταίο καιρό, η τρισδιάστατη εκτύπωση, τείνει να συγκεντρώνει όλο και περισσότερο το ενδιαφέρον των προγραμματιστών και όχι μόνο. Φυσικά, δεν θα μπορούσε κάτι τέτοιο να γίνει εφικτό χωρίς τη χρήση μικροελεγκτών. Η παραπάνω εικόνα απεικονίζει τη χρήση τριών συστημάτων Aduino UNO για την κατασκευή ενός τέτοιου συστήματος τρισδιάστατης εκτύπωσης με τη χρήση CNC.

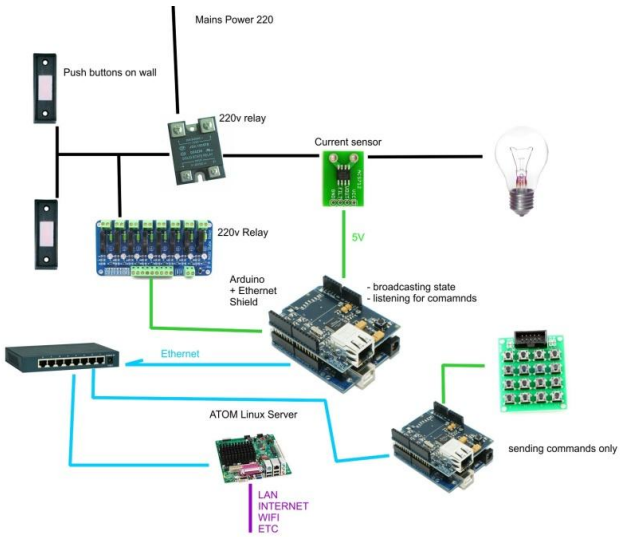
2.8.3 Ρομπότ

Σχήμα 2-18 Arduino Robot

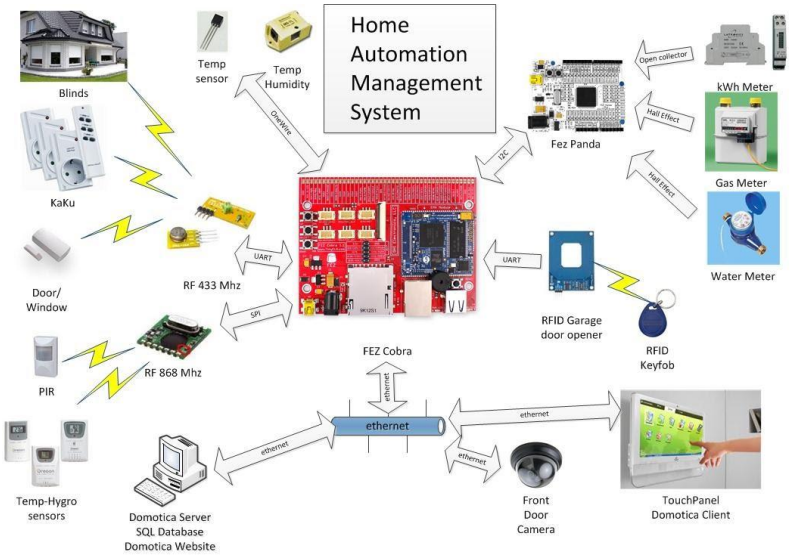
Τι πιο εύκολο από την κατασκευή ενός ρομπότ με τη βοήθεια των μικροελεγκτών? Με έναν αριθμό αισθητήρων (απόστασης, γυροσκοπικούς) και έναν αριθμό από μοτέρ μπορεί κανείς να κατασκευάσει ένα όχημα που θα μπορεί να περιφέρεται στο σπίτι αυτόνομα χωρίς να σκουντουφλάει στους τοίχους.

2.8.4 Έξυπνο σπίτι

Το έξυπνο σπίτι, δεν είναι κάτι που ακούμε πρώτη φορά. Για πολλά χρόνια περιφέρεται αυτή η έννοια, η οποία περιέγραφε ένα «de facto» ακριβό και δυσπρόσιτο σύστημα, περίπλοκο στη χρήση και κατασκευή.



Σχήμα 2-19 Arduino Smart Home

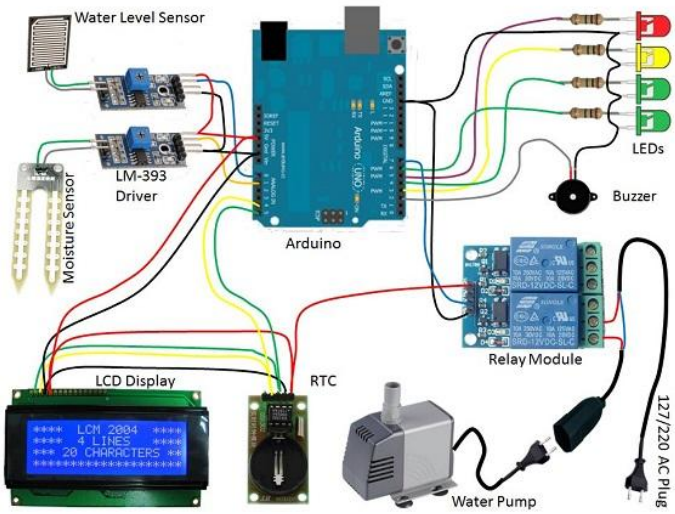


Σχήμα 2-20 Fez Smart Home System

Οι παραπάνω εικόνες απεικονίζουν συστήματα διαχείρισης έξυπνου σπιτιού βασισμένα σε ενοποιημένα συστήματα μικροελεγκτών. Με τη χρήση βοηθητικών περιφερειακών μονάδων, τα συστήματα παίρνουν μετρήσεις, αντιδρούν και αλληλεπιδρούν με το περιβάλλον και το χρήστη.

2.8.5 Αυτόματο πότισμα

Το πότισμα των λουλουδιών ή του χωραφιού, ήταν πάντα μεγάλο ζήτημα και η ανάγκη για λύση, πολύ μεγάλη. Προφανώς όταν μιλάμε για οποιονδήποτε αυτοματισμό, και την λήψη «αποφάσεων» βάσει μετρήσεων, πάντα έρχονται στο μυαλό οι μικροελεγκτές. Με ένα ενοποιημένο σύστημα μικροελεγκτή, μπορούμε να κατασκευάσουμε ένα σύστημα αυτόματου ποτίσματος, ακριβώς στα μέτρα των αναγκών του χρήστη. Με την αμέτρητη ποικιλία αισθητήρων και μονάδων και με την φαντασία των προγραμματιστών, εφόσον η δυνατότητα προγραμματισμού και παραμετροποίησης είναι ανοικτή και ελεύθερη, οι δυνατότητες οι οποίες δημιουργούνται είναι απεριόριστες. Παρακάτω φαίνεται μία μικρή υλοποίηση αυτόματου ποτίσματος.



Σχήμα 2-21 Arduino Automatic Plant Watering

2.9 Χρησιμότητα και Χρηστικότητα

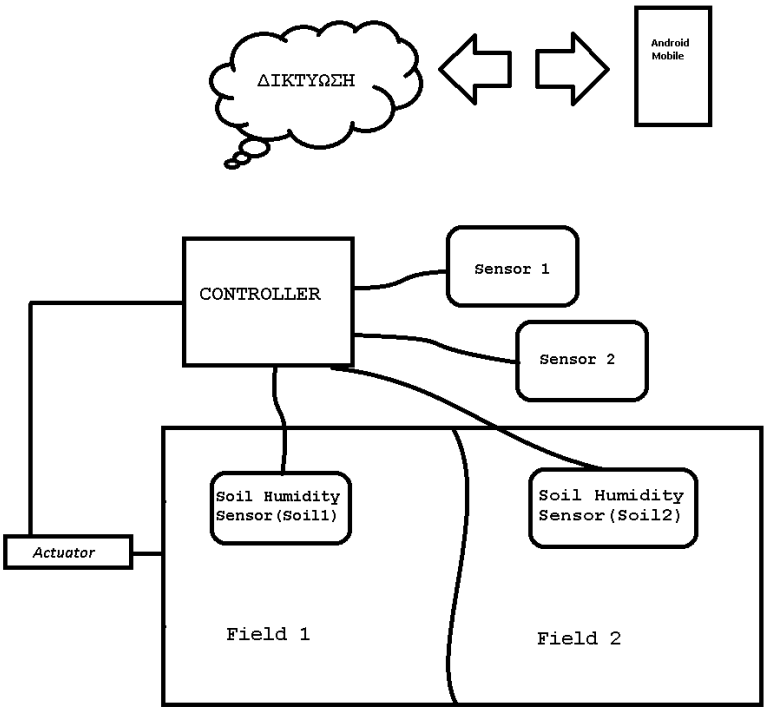
Όπως προαναφέραμε, οι δυνατότητες που δημιουργούνται, είναι απεριόριστες. Τα προβλήματα που μπορούν να αντιμετωπίσουμε και να λύσουμε είναι αμέτρητα. Παρόλα αυτά, ο σχεδιασμός και η κατασκευή τέτοιων συστημάτων δεν είναι τελείως απλή διαδικασία. Απαιτεί εξειδικευμένες γνώσεις ηλεκτρονικών, και προγραμματισμού. Περεταίρω, οι περιφερειακές μονάδες, δεν είναι κατασκευασμένες για μαζική εμπορική χρήση και σχεδόν πάντα χρειάζονται κάποια παραμετροποίηση-βελτίωση ούτως ώστε να μπορούν να χρησιμοποιηθούν στον πραγματικό κόσμο με πραγματικές συνθήκες. Ακόμα, η συναρμολόγηση και ο τελικός έλεγχος απαιτούν αρκετό χρόνο και υπομονή για τη σωστή τοποθέτηση των μονάδων και την επίλυση των προβλημάτων που εγείρονται κατά τον έλεγχο.

Η χρησιμότητα τέτοιων συστημάτων είναι αρκετά μεγάλη, αλλά για να μπορεί να αμβλυνθεί το εύρος των τελικών χρηστών, θα πρέπει ο σχεδιασμός κάθε συστήματος να προβλέπει ότι ο τελικός χρήστης μπορεί να μην έχει εξειδικευμένες γνώσεις. Οπότε θα πρέπει να λαμβάνει υπόψη, ότι ο τρόπος ρύθμισης και αλληλεπίδρασης με το σύστημα, θα πρέπει να είναι όσο πιο απλοποιημένος γίνεται.

3. ΚΕΦΑΛΑΙΟ 3
ΕΥΦΥΕΣ ΣΥΣΤΗΜΑ ΠΟΤΙΣΜΑΤΟΣ ΜΕ ΑΠΟΜΑΚΡΥΣΜΕΝΟ ΕΛΕΓΧΟ

3.1 Το Πρόβλημα που Καλούμαστε να Επιλύσουμε

Θεωρούμε ότι καλούμαστε να φροντίσουμε ένα χώρο, χωρισμένο στα δύο με μία παροχή νερού. Θα πρέπει το πότισμα, όχι μόνο να γίνεται αυτόματα, αλλά και να διαχειρίζεται απομακρυσμένα. Θα πρέπει το σύστημα, να διαθέτει έναν μικροελεγκτή που θα παίζει το ρόλο του controller. Επίσης, θα πρέπει να διαθέτει, μία σειρά αισθητήρων, θερμοκρασίας, υγρασίας χώματος, που θα τους διαχειρίζεται ο controller και θα περνούν στο σύστημα τις περιβαλλοντικές συνθήκες του χώρου. Το σύστημα με τη σειρά του θα πρέπει να αποφασίζει, σύμφωνα με τις μετρήσεις, δηλαδή αν το χώμα είναι ήδη υγρό ή ξηρό, αν θα πρέπει να ποτίσει ή όχι. Οι αισθητήρες αυτοί θα έχουν το ρόλο του INPUT στο σύστημα. Περεταίρω, το σύστημα για να μπορεί να ποτίσει, θα πρέπει να χρησιμοποιήσει ένα σετ από ηλεκτροβάνες οι οποίες θα είναι συνδεμένες στην παροχή του νερού και θα χειρίζονται ξεχωριστά από ένα σετ ρελέ. Ο συνδυασμός αυτών των δύο, θα παίζει το ρόλο του actuator (OUTPUT). Θα πρέπει το σύστημα να μπορεί να ενημερώνει το χρήστη για τις συνθήκες του χώρου και την κατάσταση του συστήματος, ανά πάσα στιγμή. Οπότε θα πρέπει με κάποια τεχνολογία δικτύωσης, να στέλνει τα δεδομένα στο κινητό του χρήστη. Φυσικά, θα πρέπει και ο χρήστης με το κινητό του τηλέφωνο να μπορεί να χειριστεί το σύστημα.



Σχήμα 3-1 Το προς υλοποίηση σύστημα

3.2 Έρευνα Αγοράς

Στην αγορά, υπάρχουν πολλά συστήματα αυτόματου ποτίσματος, και μερικά έχουν και τη δυνατότητα να ματαιώνουν το πότισμα σε περίπτωση βροχής ή ήδη υγρού εδάφους. Αναφορικά, μερικά από τα συστήματα αυτά είναι:

3.2.1 Hunter NODE 100

Το Hunter Node είναι ένα σύστημα ποτίσματος με 6 εξόδους και 4 προγράμματα, όπου μπορεί να συνδεθεί με ηλιακό συλλέκτη για αυτονομία και ψηφιακό αισθητήρα για βροχή ή υγρασία.

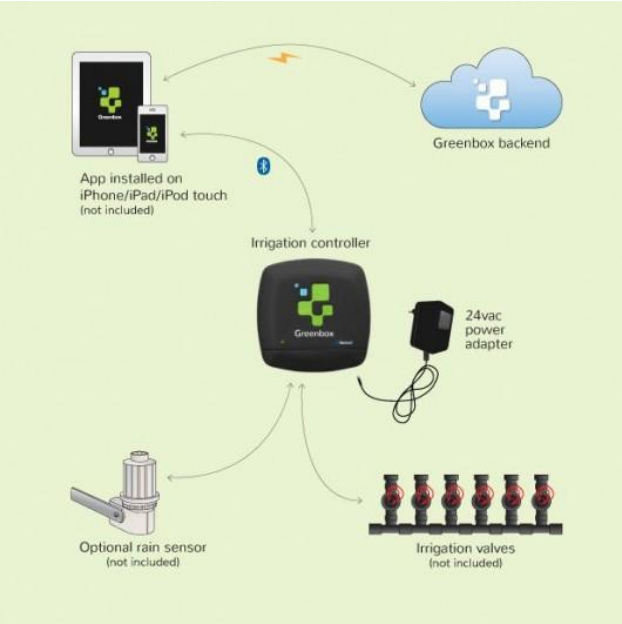
Σχήμα 3-2 Hunter Node

3.2.2 Rain Bird

Σχήμα 3-3 Rain Bird

Το Rain Bird είναι ένα σύστημα μίας εξόδου, με 6 ενάρξεις ποτίσματος.

3.2.3 GreenBox



Σχήμα 3-4 GreenBox

Το GreenBox, παρουσιάζει το μεγαλύτερο ενδιαφέρον. Είναι μία μονάδα μικροελεγκτή, που περιέχει μονάδα wifi και Bluetooth για συνδεσιμότητα, GPS για πληροφορίες τοποθεσίας, 8 εξόδους ποτίσματος και πλήρη διαχείριση μέσω Smartphone (i-phone,Android). Είναι και το μόνο που πλησιάζει σε δυνατότητες το project που παρουσιάζουμε.

3.3 Διαφορές

Η υλοποίηση που πραγματευόμαστε, αφορά αφενός ένα σύστημα απλό και συναφές με τα τυπικά υφιστάμενα συστήματα ποτίσματος, έχει όμως μία βαθμίδα ευφυΐας παραπάνω αφού μπορεί και αποφασίζει δυναμικά τη ματαίωση η μη του ποτίσματος, βασιζόμενο στις πραγματικές καιρικές συνθήκες του περιβάλλοντος του χώρου. Επιπροσθέτως, καταγράφει σε έναν απομακρυσμένο διακομιστή, τις καιρικές συνθήκες (θερμοκρασία, υγρασία, υγρασία χώματος) και την κατάσταση του συστήματος (System State). Τέλος, όλο το σύστημα παρακολουθείται και ελέγχεται μέσω ενός Smartphone.

3.4 Προβλήματα που εγείρονται

Κατά τον σχεδιασμό κάθε συστήματος, εγείρονται πάντα πολλά προβλήματα όσον αφορά, τα υλικά που θα χρησιμοποιηθούν τον τρόπο προσέγγισης του θέματος και πολλά άλλα. Στο συγκεκριμένο project, πραγματευόμαστε ένα σύστημα που θα ποτίζει βάσει κάποιου χρονοδιαγράμματος, το οποίο δύναται να αλλάξει ανάλογα τις καιρικές συνθήκες του χώρου. Ταυτόχρονα, το σύστημα θα παρακολουθείται και θα ελέγχεται από το χρήστη, με τη χρήση μίας εφαρμογής Android.

- **Τι να Διαλέξω**

  Αρχικά, το πρώτο πρόβλημα είναι το ποια οικογένεια μικροελεγκτών πρέπει να διαλέξουμε και γιατί. Η επιλογή μας θα πρέπει να βασίζεται σε κριτήρια μνήμης, απόδοσης, ποικιλία περιφερειακών υλικών και ευκολίας στον προγραμματισμό.
- **Συνδεσιμότητα**

  Θα πρέπει το σύστημά μας, να μπορεί να επικοινωνήσει με το χρήστη ανεξάρτητα από το που βρίσκεται. Διότι αν βρίσκεται σε κάποιο απομακρυσμένο αγροτεμάχιο, σπάνια υπάρχει δυνατότητα επίγειου internet. Οπότε μία μονάδα wifi ή Ethernet, πιθανότατα να μας ήταν τελείως άχρηστη.
- **Διεπαφή**

  Το σύστημα θα πρέπει να μπορεί να παραμετροποιηθεί και από την εφαρμογή, αλλά και από την ίδια τη συσκευή. Όπως προαναφέραμε λοιπόν, θα πρέπει να σχεδιάσουμε μία διεπαφή συστήματος/χρήστη για εύκολη και κατανοητή ρύθμιση των παραμέτρων του συστήματος και τον έλεγχό του.
- **Διακομιστής Καταγραφής (Data Log Server)**

  Για τις ανάγκες της καταγραφής των δεδομένων του συστήματος, χρειάζεται η εγκατάσταση ενός διακομιστή ο οποίος θα είναι προσβάσιμος από παντού και θα είναι ικανός να αποθηκεύει τα δεδομένα του συστήματος ανά πάσα στιγμή (**Apache server**, **MySQL server** κτλ). Ιδανικά, θα μπορούσε να χρησιμοποιηθεί και κάποια "cloud" υπηρεσία καταγραφής δεδομένων όπως το "**Xively".**
- **Η ώρα**

  Θα πρέπει το σύστημα κάθε στιγμή, να γνωρίζει την πραγματική ώρα. Θα πρέπει να μπορεί να την κρατάει ακόμα και σε περίπτωση που για κάποιο λόγο προβεί σε επανεκκίνηση.
- **Αποθήκευση**

  Το σύστημα, θα λειτουργεί βάση κάποιων παραδοχών και λογικών αποφάσεων. Αυτό σημαίνει ότι θα πρέπει αφενός να «θυμάται» το χρονοδιάγραμμα που του έχει οριστεί, αφετέρου όμως θα πρέπει να «θυμάται» και τις δυναμικές αλλαγές που συμβαίνουν στη διάρκεια της ημέρας.

4. ΚΕΦΑΛΑΙΟ 4

ΥΛΟΠΟΙΗΣΗ ΣΥΣΤΗΜΑΤΟΣ

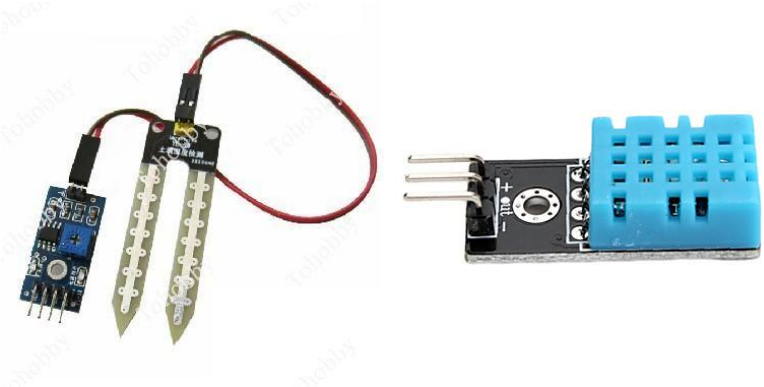
4.1 Εξοπλισμός και Εξήγηση

- **Αναπτυξιακή πλακέτα**
  Η γκάμα και η ποικιλία όσον αφορά τις αναπτυξιακές πλατφόρμες είναι πάρα πολύ μεγάλη και οι επιλογές πάρα πολλές. Για το παρόν project διαλέξαμε την αναπτυξιακή πλατφόρμα της οικογένειας Arduino Mega2560 που ενσωματώνει τον μικροελεγκτή Atmel ATmega2560. Κριτήρια για αυτήν την επιλογή ήταν αρχικά η ευρύτητα της αποδοχής που τυγχάνει η συγκεκριμένη οικογένεια (Arduino) ενοποιημένων συστημάτων μικροελεγκτών. Δευτερευόντως, η συγκεκριμένη πλακέτα διαθέτει ικανοποιητικό αριθμό I/O θηρών και αρκετά μεγάλο μέγεθος μνημών (Flash, SRAM,EEPROM) ώστε να καθιστά εφικτή την υλοποίηση μίας εφαρμογής αυτού του μεγέθους.
- **Συνδεσιμότητα**
  Για μπορεί το σύστημα να αλληλεπιδράσει με το χρήστη και να καταγράψει τα δεδομένα που παράγει, θα πρέπει να χρησιμοποιήσουμε μία περιφερειακή μονάδα διασύνδεσης του συστήματος με τον πραγματικό κόσμο (Internet). Επίσης οι επιλογές είναι αρκετές (ethernet, wifi, GSM/GPRS, Bluetooth, RF αλλά με γνώμονα το ότι η εφαρμογή μας θα πρέπει να μπορεί να τοποθετηθεί οπουδήποτε, χωρίς την μη ύπαρξη ασύρματου ή ενσύρματου internet να μπαίνει εμπόδιο στη σωστή και ολοκληρωμένη λειτουργία του, καταλληλότερη μονάδα διασύνδεσης κρίναμε ότι είναι μία μονάδα GSM/GPRS. Με αυτόν τον τρόπο, το σύστημα θα έχει Internet παντού χωρίς την ανάγκη επίγειας γραμμής, και θα μπορεί να αλληλεπιδρά και μέσω SMS.

Σχήμα 4-1 GPRS shield

- **Καιρικές Συνθήκες**
  Για να μπορεί το σύστημα να γνωρίζει τις καιρικές συνθήκες του περιβάλλοντος του και να μπορεί να παίρνει αποφάσεις, θα πρέπει να χρησιμοποιήσει και τους ανάλογους αισθητήρες. Αρχικά, θα χρησιμοποιήσουμε την οικογένεια ενοποιημένων αισθητήρων DHTxx για τη μέτρηση θερμοκρασίας και υγρασίας περιβάλλοντος. Συγκεκριμένα θα χρησιμοποιήσουμε τον αισθητήρα DHT11. Περεταίρω, μία ακόμα

μέτρηση που θα πρέπει να παίρνει το σύστημα, είναι η υγρασία του χώματος. Στην πραγματικότητα, είναι και η σημαντικότερη μέτρηση του συστήματος. Η υγρασία του χώματος είναι ενδεικτική της πραγματικής ανάγκης του χώρου για πότισμα ή όχι. Για αυτή τη μέτρηση, θα χρησιμοποιήσουμε έναν αισθητήρα υγρασίας χώματος με αναλογική έξοδο.



Σχήμα 4-2 Arduino compatible Soil Moisture Sensor / Temperature-Humidity Sensor DHT11

- **Η ώρα**

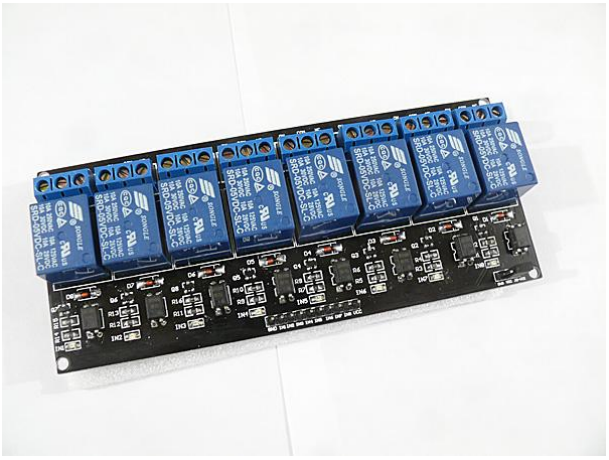
  Όπως προαναφέραμε, το σύστημα θα πρέπει να συγκρατεί την πραγματική ώρα, ακόμα και αν το σύστημα προβεί σε επανεκκίνηση. Αυτό σημαίνει ότι θα πρέπει να χρησιμοποιήσουμε μία μονάδα RTC η οποί θα διαθέτει μπαταρία για αυτονομία δυνατότητας συγκράτησης της ώρας σε όλες τις περιπτώσεις.

Σχήμα 4-3 Arduino compatible RTC (Real Time Clock) module

- **Έλεγχος**

Αφού το σύστημα πάρει τις απαραίτητες λογικές αποφάσεις του, θα πρέπει προβεί σε κάποιες κινήσεις ελέγχου. Θα πρέπει δηλαδή, να ξεκινήσει ή να σταματήσει το πότισμα. Αυτό απορεί να γίνει εφικτό με τη χρήση ηλεκτρονόμων (Ρελέ/Relay) ισχύος και ηλεκτροβανών. Το σύστημα θα δίνει την εντολή HIGH (5v) ή LOW(0v) και ο ηλεκτρονόμος θα «οπλίζεται» ή θα «αφοπλίζεται» και θα δρομολογεί τη θετική τάση στην ηλεκτροβάνα ούτως ώστε να ανοίξει και να ποτίσει, ή θα αποκόπτει την τάση και θα κλείνει την ηλεκτροβάνα διακόπτοντας το πότισμα. Για αυτή τη

λειτουργία, θα χρησιμοποιήσουμε μία ενοποιημένη περιφερειακή μονάδα ηλεκτρονόμων με ενσωματωμένο κύκλωμα προστασίας, και μία ηλεκτροβάνα διέγερσης 12v.



Σχήμα 4-4 Arduino Compatible relay Module

Σχήμα 4-512v Electrovalve

- **Διεπαφή**

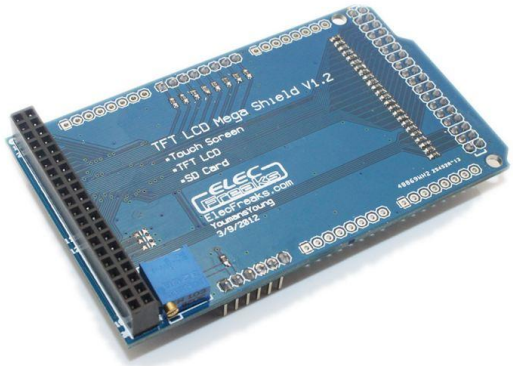
  Όπως προαναφέραμε σε προηγούμενο κεφάλαιο, θα πρέπει ο τελικός χρήστης να μπορεί εύκολα να χειριστεί και να παραμετροποιήσει το σύστημα. Αντίθετα με τα υφιστάμενα συστήματα, το σύστημα μας δεν θα διαθέτει δυσνόητα σχεδιαγράμματα και κουμπιά.

  Αντιθέτως, ο πιο προσιτός και απλοϊκός τρόπος αλληλεπίδρασης, είναι ένα εύκολο μενού προσπελάσιμο από μία οθόνη αφής. Πιο συγκεκριμένα, θα χρησιμοποιήσουμε μία έγχρωμη 3.2” οθόνη αφής.

  Σχήμα 4-63.2” touch screen module

  Όσον αφορά τον απομακρυσμένο έλεγχο και αλληλεπίδραση, όπως αναφέραμε παραπάνω, θα επιτευχθεί μέσω μίας εφαρμογής-διεπαφής βασισμένης στην πλατφόρμα **Android.**

- **Αποθήκευση**
  Αυτό το κομμάτι είναι πολύ σημαντικό για τη σωστή λειτουργία του συστήματος. Θα πρέπει το σύστημα να μπορεί να αποθηκεύει τις αρχικές του ρυθμίσεις και το εκάστοτε χρονοδιάγραμμα. Αυτό μπορεί να γίνει εφικτό με τη χρήση μίας μονάδας εξωτερικής αποθήκευσης (SDcard). Στη συγκεκριμένη περίπτωση, θα χρησιμοποιήσουμε μία περιφερειακή μονάδα, η οποία είναι συμβατή με την οθόνη αφής. Αυτή η μονάδα σε συνδυασμό με την οθόνη υλοποιούν ένα ενοποιημένο σύστημα απεικόνισης και αποθήκευσης.



Σχήμα 4-7 TFT SD card shield for Arduino

4.2 Προγραμματιστικά Εργαλεία

Εκτός από τη συλλογή των κατάλληλων περιφερειακών μονάδων και τη συναρμολόγηση τους, για να μπορέσει το σύστημα να εκτελέσει τις απαραίτητες λειτουργίες για τις οποίες προορίζεται, απαιτείται μακροσκελής και εξειδικευμένος προγραμματισμός. Η «γλώσσα» μπορεί να ποικίλει ανάλογα το υλικό που έχουμε χρησιμοποιήσει.

- **Arduino IDE**
  Για τον προγραμματισμό και παραμετροποίηση της εφαρμογής μας, χρησιμοποιήσαμε το εργαλείο ανάπτυξης της ίδιας της Aduino. Είναι μία ανοικτού κώδικα (open source) πλατφόρμα ανάπτυξης, «μετάφρασης» και μεταφόρτωσης. Η πλατφόρμα είναι «γραμμένη» σε Java, και χρησιμοποιεί τη γλώσσα C++ για τον προγραμματισμό των μικροελεγκτών(Atmel). Φυσικά, οι

μικροελεγκτές δεν «καταλαβαίνουν» αυτή τη γλώσσα, αλλά τη γλώσσα μηχανής(Assembly). Το ρόλο του μεταφραστή τον αναλαμβάνει η ίδια η πλατφόρμα. Έπειτα, μέσω USB αναλαμβάνει και τη μεταφόρτωση του κώδικα από τον Η/Υ στην αναπτυξιακή πλακέτα.

Σχήμα 4-8 Arduino Open Source Community

- **SSCOM**

  Κατά τη διάρκεια του προγραμματισμού, πολλές φορές ο προγραμματιστής αντιμετωπίζει κάποια σφάλματα ή κάποια αμφιλεγόμενα αποτελέσματα τα οποία προέρχονται ή από εσφαλμένο σχεδιασμό, ή από κάποιο προγραμματιστικό

  λάθος.

  Για τη διόρθωση, ο προγραμματιστής χρειάζεται κάποια εργαλεία παρακολούθησης και αποσφαλμάτωσης. Στον προγραμματισμό μικροελεγκτών, μία προσφιλής τακτική, είναι εμφάνιση μηνυμάτων στη σειριακή θύρα. Με αυτόν τον τρόπο, ο προγραμματιστής ελέγχει την ορθή λειτουργία της εφαρμογής του. Όμως, για να μπορέσει να παρακολουθήσει την έξοδο της σειριακής πόρτας της αναπτυξιακής πλακέτας, χρειάζεται ένα ανάλογο εργαλείο. Η πλατφόρμα Arduino IDE διαθέτει ενσωματωμένο ένα τέτοιο εργαλείο. Όμως για τις πιο εξειδικευμένες ανάγκες της εφαρμογής μας, χρησιμοποιήσαμε την εφαρμογή **SSCOM.** Με την οποία μπορούμε να παρακολουθούμε την έξοδο της σειριακής θύρας και ταυτόχρονα να αλληλεπιδρούμε με τη συσκευή, με διάφορους τρόπους.

- **Eclipse**

  Για τον απομακρυσμένο έλεγχο και αλληλεπίδραση με το σύστημά μας, θα χρησιμοποιήσουμε μία εφαρμογή για Smartphone. Εμείς, επιλέξαμε την

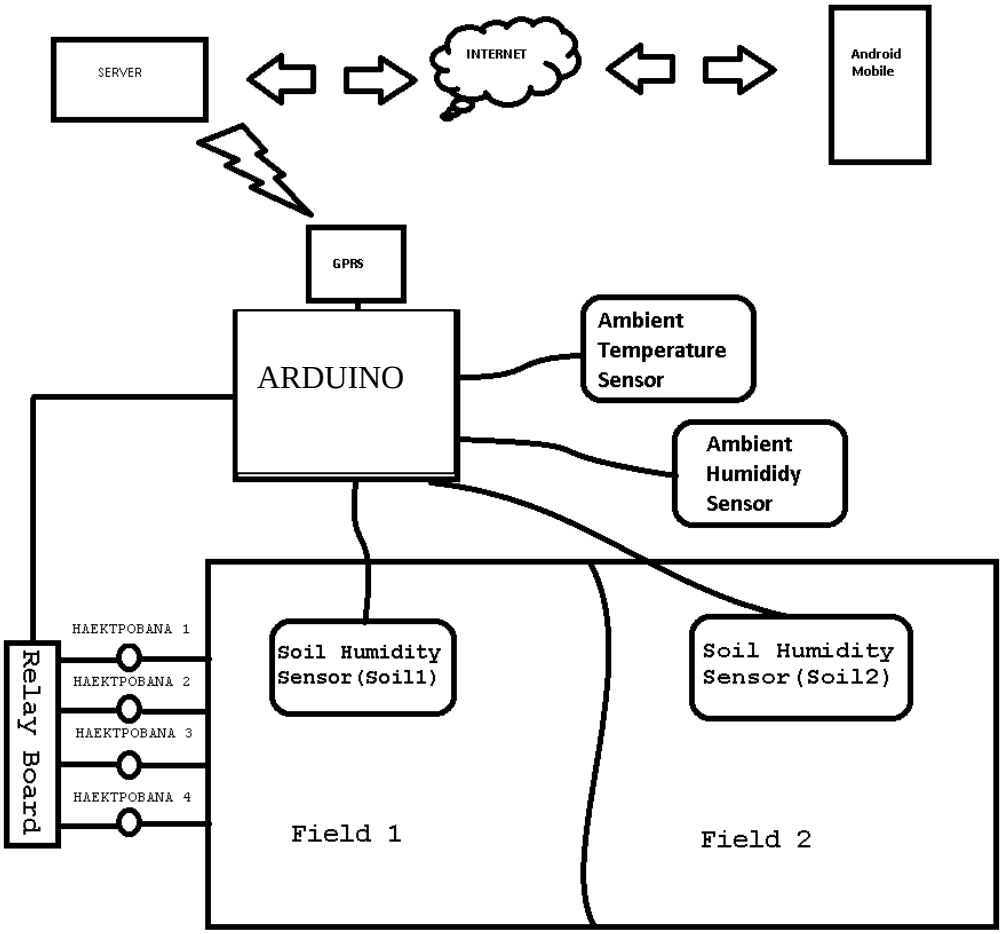
πλατφόρμα Android η οποία είναι ευρέως διαδεδομένη, και τα προγραμματιστικά της εργαλεία, δωρεάν.
Για την δημηουργία της εφαρμογής αυτής, θα χρησιμοποιήσουμε την πλατφόρμα ανάπτυξης **eclipse.** Η eclipse, είναι εργαλείο ανάπτυξης για τη γλώσσα **java,** όμως μπορεί να ενσωματώσει το εργαλείο και τις βιβλιοθήκες ανάπτυξης android.

Σχήμα 4-9 The eclipse project

Η πλατφόρμα διαθέτει επίσης και emulator για τη δοκιμή της εφαρμογής σε πραγματικό χρόνο, χωρίς να χρειάζεται να μεταφορτώνεται κάθε φορά σε κάποιο Smartphone.

4.3 Σχεδιασμός

Ο αρχικός σχεδιασμός, προέβλεπε ένα σύστημα ποτίσματος με περιορισμένες δυνατότητες, που θα μπορούσε να καλύψει μόνο οικιακές ανάγκες, όπως πχ πότισμα γλαστρών η μικρού κήπου. Αφενός, τέτοια παρόμοια συστήματα υπάρχουν ήδη στο εμπόριο, αφετέρου έχουμε τη δυνατότητα να κατασκευάσουμε κάτι που θα καλύψει τις πραγματικές μας ανάγκες, σε οποιαδήποτε κλίμακα χρειαστεί. Οπότε καταλήξαμε στο παρακάτω αποτέλεσμα.



Σχήμα 4-10 Project Plan

4.3.1 Συνδεσιμότητα

Για να μπορεί το σύστημα να εγκατασταθεί σε οποιοδήποτε χώρο, που ενδεχομένως να μην μπορεί να έχει επίγειο internet, θα πρέπει να μπορεί να συνδεθεί στο διαδίκτυο μέσω του δικτύου κινητής τηλεφωνίας(GPRS/3G/2G/GSM). Αυτό είναι και ένα από τα στοιχεία που καθιστούν το σύστημά μας διαφορετικό από όλα τα υπόλοιπα. Για την επίλυση ουτού του προβλήματος θα χρησιμοποιήσουμε ένα **Arduino compatible 3G/GPRS shield (Σχ. 4.1)**. Ταυτόχρονα, με τη χρήση αυτής της περιφεριακής μονάδας, θα μπορεί να υπάρχει αλληλεπίδραση με το χρήστη μέσω SMS.

4.3.2 Καταγραφή

Το σύστημά μας μπορεί εν δυνάμη να εγκατασταθεί σε κάποιο απομακρισμένο σημείο, όπου η δυνατότητα επίσκεψης να είναι περιορισμένη. Το γεγονός αυτό, καθιστά την ανάγκη για επίβλεψη (θερμοκρασία, υγρασία κτλ) επιτακτική. Αυτό μπορεί να επιτευχθεί με διάφορους τρόπους. Η υλοποίηση που διαλέξαμε, είναι η χρήση ενός απομακρυσμένου **MySQL server** όπου το σύστημα μας θα καταγράφει τα δεδομένα του περιβάλοντος του και και κάποιες άλλες πληροφορίες που αφορούν την κατάσταση του ίδιου του συστήματος. Φυσικά αυτές οι πληροφορίες θα μπορούν να προσπελαστούν από παντού. Επιπλέον με την διαδοχική καταγραφή των καιρικών συνθηκών, θα μπορούμε μελλοντικά να εξαγάγουμε και κάποια χρήσιμα στατιστικά συμπεράσματα.

4.3.3 Αλληλεπίδραση

Το σύστημα, για να μπορέσει να προσαρμοστεί στις ανάγκες μας, θα πρέπει αρχικά να υποστεί έναν χρονοπρογραμματισμό που θα ορίζει τις όρες ποτίσματος. Τα μέχρι τώρα υφιστάμενα συστήματα, διαθέτουν δυσνόητες διεπαφές που με πολύ προσπάθεια κανείς καταλαβαίνει πως λειτουργούν. Μέλιμα μας στον σχεδιασμό, είναι η όσο το δυνατό ευκολότερη αλληλεπίδραση με το σύστημα. Για αυτό το λόγο χρησημοποιήσαμε μία **οθόνη αφής 3.2" (Σχ. 4.6)** η οποία καθιστά τον χειρισμό και προγραμματισμό του συστήματος πολύ απλό για οποιονδήποτε χρήστη. Επιπλέον, θα πρέπει η επίβλεψη και ο έλεγχος, να είναι εξίσου απλός για το χρήστη. Για αυτό το λόγο, αναπτύχθηκε μία εφαρμογή για την πλατφόρμα **Android** η οποία θα έχει τη δυνατότητα να παρουσιάζει τα καταγεγραμένα δεδομένα στον απομακρυσμένο **MySQL server** και να αξιοποιεί τα σταστιστικά δεδομένα. Ακόμι θα μπορεί να σταματά και να ξεκινά τη λειτουργεία του συστήματος κατά το δοκούν.

5. ΚΕΦΑΛΑΙΟ 5

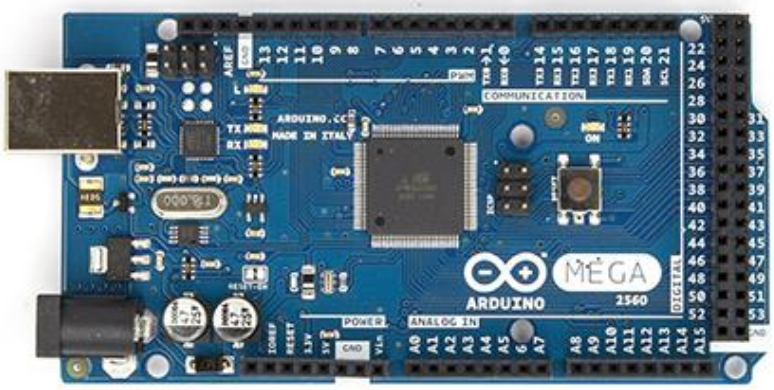
ΣΥΝΔΕΣΜΟΛΟΓΙΑ

5.1 Γενικά

Η παρούσα υλοποίηση, πραγματεύεται ένα σύστημα με δύο ηλεκτροβάνες οι οποίες θα μπορούν να χειριστούν 2 μεγάλους ξεχωριστούς χώρους (Χωράφια). Ο κάθε χώρος έχει έναν δικό του αισθητήρα υγρασίας χώματος σύμφωνα με τις μετρήσεις του οποίου, το σύστημα θα παίρνει αποφάσεις όσον αφορά το πότισμα.
Αναλυτικότερα το σύστημα και τα περιφερειακά που το αποτελούν περιγράφονται ακολούθως.

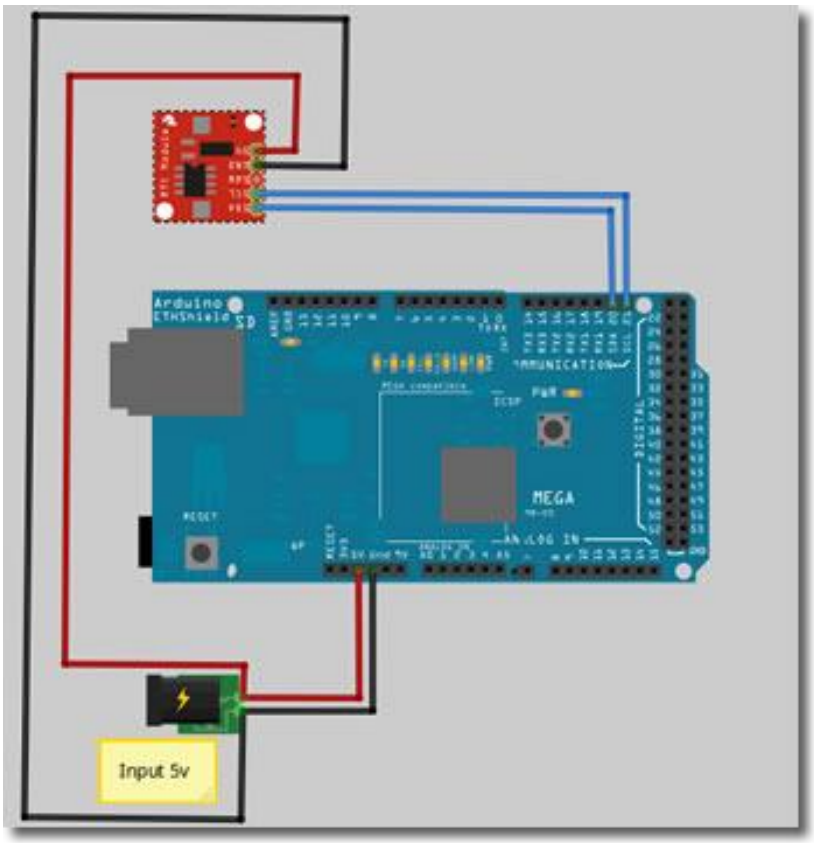
5.2 Αναπτυξιακή Πλακέτα

Η αναπτυξιακή πλακέτα που χρησιμοποιούμε είναι η **Arduino Mega2560** όπου ενσωματώνει τον μικροελεγκτή ATMEL ATmega2560**.** Η τροφοδοσία της είναι από 9 έως 12 V, διαθέτει 54 ψηφιακές θήρες(Input/Output), εκ των οποίων οι 15 μπορούν να χρησιμοποιηθούν σαν PWM θήρες, 16 αναλογικές εισόδους και 4 UARTs(σειριακές θήρες επικοινωνίας) και η από κατασκευής διασύνδεση της με τον Η/Υ είναι η θύρα USB η οποία μεταφέρει τα δεδομένα της σειριακής διεπαφής του συστήματος με έναν οδηγό (Driver) που παρέχει η ίδια η εταιρεία όπου προσομοιώνει την θύρα USB με σειριακή στον Η/Υ. Έτσι με ένα οποιοδήποτε **Serial Monitor** μπορούμε να παρακολουθούμε τα μηνύματα που παράγει κατά τη διαδικασία πράγμα που μας βοηθά στην επίλυση τυχόν προβλημάτων, αλλά και γενικότερα στην αλληλεπίδραση με το σύστημα, αφού η σειριακή διεπαφή είναι αμφίδρομη.



5.2.1 Real Time Clock

Όπως αναφέρεται σε παραπάνω κεφάλαιο, θα πρέπει το σύστημα να μπορεί να ξέρει την σωστή ώρα ανά πάσα στιγμή ανεξάρτητα αν κλείσει ή επανεκκινηθεί. Το Real Time Clock, είναι ένα σύστημα, το οποίο ενσωματώνει ένα τσιπ χρονοϋπολογισμού που ανεξάρτητα από την τοπική τροφοδοσία της πλακέτας, χρησιμοποιεί και μία εξωτερική πηγή ενέργειας (μπαταρία) για να μπορεί να κρατάει τα δεδομένα του ανεξάρτητα από την κατάσταση του συστήματος.



Σχήμα 5-1 Σύνδεση Arduino με RTC

Το **RTC** παίρνει τροφοδοσία 5V και **GND** τα οποία παίρνουμε από την πλακέτα, πράγμα που δεν επηρεάζει τη γενική τροφοδοσία αφού η ισχύς που καταναλώνει είναι αμελητέα. Περεταίρω χρειάζεται άλλες δύο συνδέσεις για τη σωστή λειτουργία του. Η μία είναι η **SDA** (Serial Data Input/Output), είναι δηλαδή η σειριακή σύνδεση της συσκευής με το σύστημα μας και στην πλακέτα μας είναι η αφιερωμένη(dedicated) η ψηφιακή πόρτα **20** . Και η άλλη είναι η **SCL** (Serial Clock Input), είναι δηλαδή η είσοδος για το τσιπ του χρονοϋπολογισμού (DS1307). Σε αυτήν την πόρτα στέλνει το πρόγραμμα μας τα δεδομένα συγχρονισμού της ώρας.

5.2.2 Πλακέτα Ρελέ NO/NC

Για να μπορεί το σύστημα μας να ελέγχει το πότισμα των χώρων, θα πρέπει να ανοιγοκλείνει 2 ηλεκτροβάνες με τάση ενεργοποίησης 12V. Αυτό φυσικά δεν μπορεί η αναπτυξιακή μας πλακέτα να το διεκπεραιώσει μόνη της, αφού στην έξοδό της, μπορεί να παράγει μόνο έως 5V. Για να το καταφέρουμε αυτό πρέπει να χρησιμοποιήσουμε 2 ρελέ ισχύος με τάση ενεργοποίησης 5V, και μία εξωτερική πηγή τροφοδοσίας 12V για τις ηλεκτροβάνες. Ο συνδυασμός αυτών των δύο λαμβάνει το ρόλο του **Actuator** στο σύστημα .

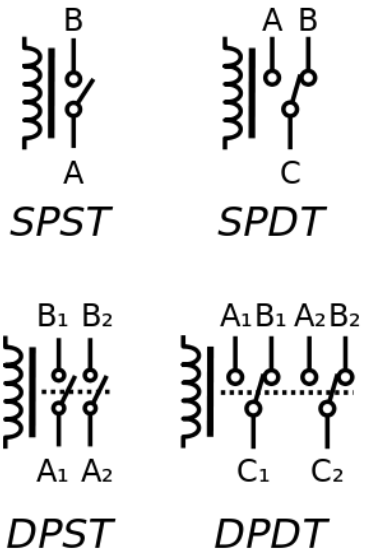
5.2.3 Ηλεκτροβάνες

Στην υλοποίηση μας χρησιμοποιούμε ένα σετ ηλεκτροβανών, με τάση διέγερσης 12V και νορμάλ λειτουργία NO(Normal Open), δηλαδή με ανοικτό κύκλωμα και η ισχύς που καταναλώνει είναι 250mA.

Σχήμα 5-2 Ηλεκτροβάνα

5.2.4 Ρελέ

Το ρελέ, είναι ένας ηλεκτρονικά χειριζόμενος διακόπτης. Περιέχει ένα πηνίο, που όταν διεγερθεί ηλεκτρικά μαγνητίζει έναν ακροδέκτη του κυκλώματος, και έτσι ένας κινητός ακροδέκτης κολλά ή αποκολλά από επάνω του κλείνοντας ή ανοίγοντας το κύκλωμα.
Υπάρχουν 4 γενικές κατηγορίες ρελέ, αλλά εμείς θα χρησιμοποιήσουμε την πρώτη **SPST(Single Pole Single Throw).**



Σχήμα 5-3 Relay Types

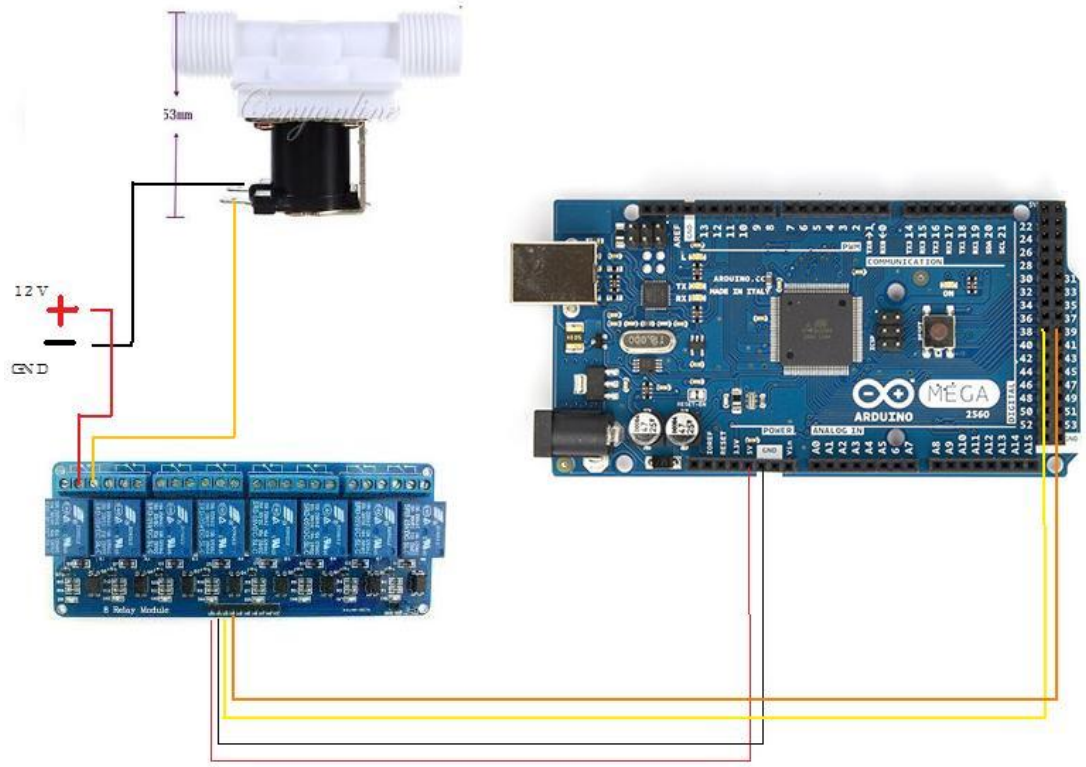
5.2.5 Arduino Relay Module

Για κατασκευαστική ευκολία θα χρησιμοποιήσουμε μία έτοιμη κατασκευή, συμβατή με την αναπτυξιακή μας πλακέτα, η οποία ενσωματώνει και τους ακροδέκτες σύνδεσης και τα ενδεικτικά **LED** λειτουργίας.

Σχήμα 5-4 Relay Module

Η εν λόγω πλακέτα, έχει τη δυνατότητα διασύνδεσης 8 ρελέ, αλλά στην παρούσα υλοποίηση θα χρησιμοποιήσουμε 2. Ομοίως με το RTC τροφοδοτείται με 5V και GND απευθείας από την αναπτυξιακή πλακέτα. Έπειτα τα ρελέ 1 και 2, συνδέονται στις ψηφιακές θήρες 48 και 49 αντίστοιχα. Στον κοινό ακροδέκτη του ρελέ, συνδέουμε το το VCC (12V) της εξωτερικής τροφοδοσίας και στον ακροδέκτη NC (κανονικά κλειστό) συνδέουμε τον ένα ακροδέκτη της

ηλεκτροβάνας. Τέλος, στον εναπομείναντα ακροδέκτη της ηλεκτροβάνας, συνδέουμε το GND της εξωτερικής τροφοδοσίας.

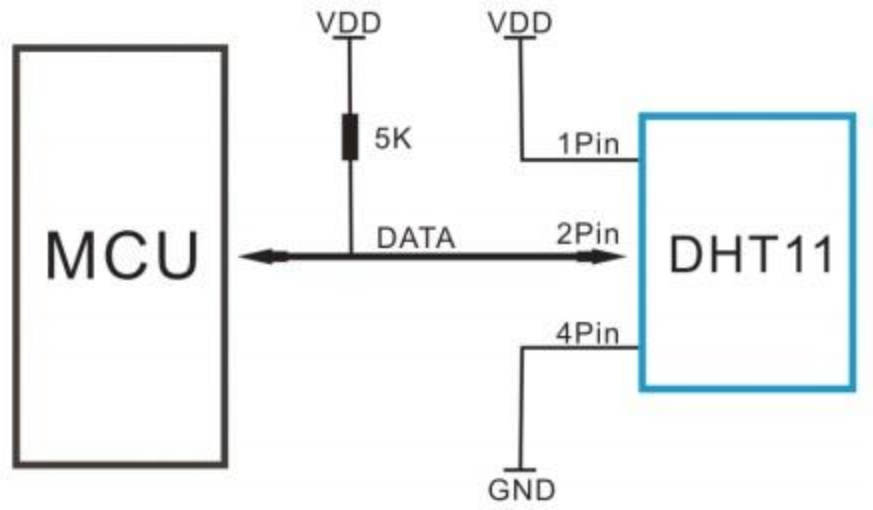


Σχήμα 5-5 Σύνδεση Arduino με relay module και ηλεκτροβάνας

5.3 Αισθητήρες

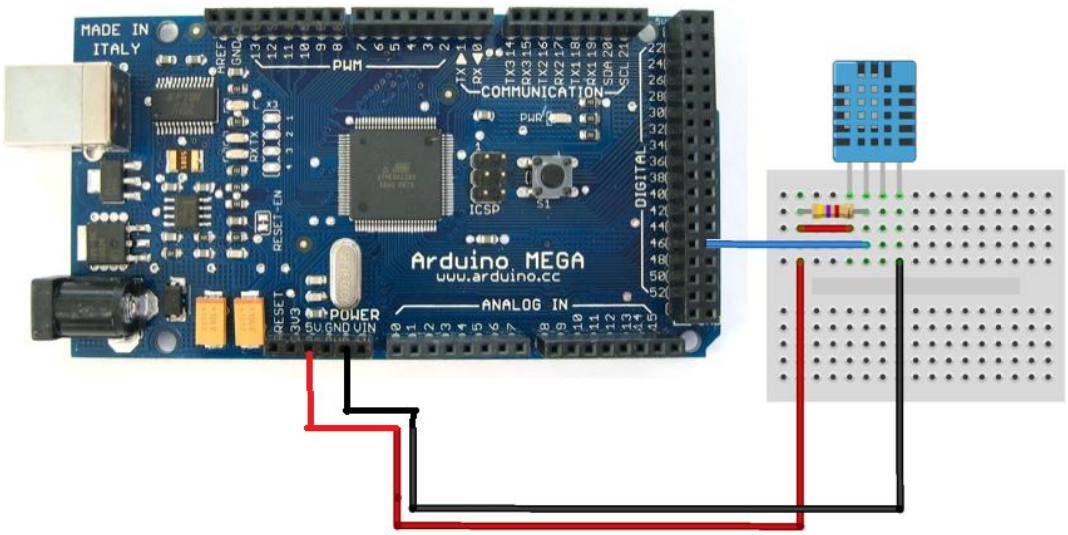
5.3.1 Θερμοκρασία/Υγρασία

Για τη μέτρηση της θερμοκρασίας και της υγρασίας, χρησιμοποιήθηκε ένας αισθητήρας διπλής ιδιότητας, ο DHT11 ο οποίος μετρά ταυτόχρονα και θερμοκρασία και υγρασία περιβάλλοντος. Ο DHT διαθέτει 4 pins εκ των οποίων μας χρησιμεύουν τα 3 για τη σύνδεση και λειτουργία του. Αναλυτικότερα:



Σχήμα 5-6 Αισθητήρας DHT11

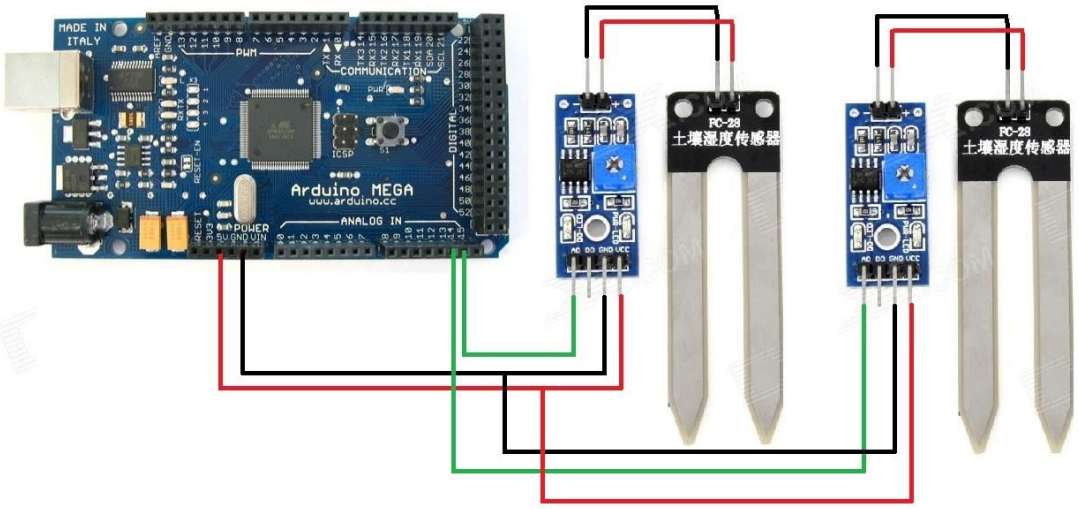
Παρατηρούμε ότι στο 1^{ο} pin συνδέουμε την τάση (VDD 3-5.5V) απευθείας από την πλακέτα αφού η μέγιστη ζήτηση ισχύς του είναι αμελητέα (2.5mA). Στο 2^{ο} pin συνδέουμε τον αισθητήρα με μία ψηφιακή είσοδο της πλακέτας και στην προκειμένη περίπτωση το συνδέσαμε στην ψηφιακή θύρα 47, και σύμφωνα με τον κατασκευαστή, παράλληλα με αυτό συνδέσαμε τη τάση(VDD) με μία αντίσταση της τάξης των 5 ΚΩ. Το 3^{ο} pin δεν χρησιμοποιείται. Τέλος στο 4^{ο} pin συνδέουμε το GND.



Σχήμα 5-7 Συνδεση Arduino με DHT11

5.3.2 Υγρασία Χώματος

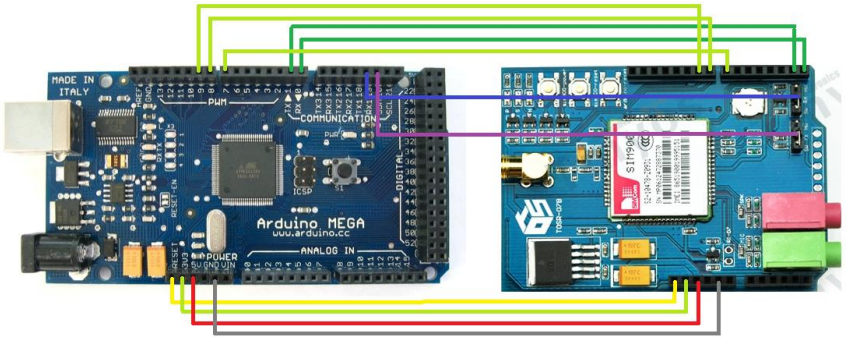
Για τη μέτρηση της υγρασίας του χώματος, χρησιμοποιήσαμε μία περιφερειακή συσκευή, συμβατή με την αναπτυξιακή μας πλακέτα, η οποία ενσωματώνει εκτός από τον αισθητήρα και ρυθμιστή ευαισθησίας. Αυτός ο ρυθμιστής μας χρησιμεύει στη βαθμονόμηση της συσκευής που χρειάζεται να γίνεται περιοδικά. Η συσκευή αποτελείται από δύο ακροδέκτες, οι οποίοι προεκτείνονται και καταλήγουν σε δύο πιο πλατύς ακροδέκτες οι οποίοι είναι αυτοί που μετρούν την αντίσταση του εδάφους και μας επιστρέφει το ποσοστό της υγρασίας που περιέχεται τοπικά της μέτρησης.
Ακόμη διαθέτει άλλους 4 ακροδέκτες, εκ των οποίων ο πρώτος(αριστερά προς τα δεξιά), είναι η αναλογική έξοδος της συσκευής μέσω της οποίας παρακολουθούμε αναλυτικά τις μετρήσεις των ακροδεκτών , ο δεύτερος είναι η ψηφιακή έξοδος της συσκευής η οποία σαν έξοδο μας παρέχει δύο αποτελέσματα(HIGH/LOW) δηλαδή αν το χώμα είναι υγρό ή όχι, ο τρίτος είναι η γείωση της συσκευής(GND) και ο τέταρτος, είναι για την παροχή της τάσης του συστήματος. Τάση και γείωση τα παρέχουμε από την πλακέτα αφού ομοίως με τις παραπάνω συσκευές, η ισχύς που καταναλώνει το κύκλωμα είναι 35mA. Στην παρούσα υλοποίηση για την αναλυτικότερη απεικόνιση των μετρήσεων θα χρησιμοποιήσουμε την αναλογική έξοδο του κυκλώματος. Ακόμα για τις ανάγκες του συστήματος, θα χρησιμοποιήσουμε 2 δύο τέτοια συστήματα τα οποία θα συνδέσουμε στις αναλογικές θήρες της πλακέτας A15 και A14 αντίστοιχα.



Σχήμα 5-8 Σύνδεση Arduino με αισθητήρα υγρασίας χώματος

5.4 GSM/GPRS

Για τη διασύνδεση του συστήματος με τον έξω κόσμο (internet) και για την αλληλεπίδραση του με τον χρήστη θα χρησιμοποιήσουμε μία περιφερειακή μονάδα GSM/GPRS η οποία ενσωματώνει το τσιπ **SIM900** της **SIMCom** και με τη χρήση των δικτύων κινητής τηλεφωνίας GSM και του δικτύου $2^{ης}$ και $3^{ης}$ γενιάς (2G/3G) αποκτά τη λειτουργικότητα και τις δυνατότητες ενός κινητού τηλεφώνου. Μπορεί να συνδέεται στο internet, να κάνει HTTP client αιτήματα, να κάνει και να δέχεται κλήσεις όπως και να στέλνει και να δέεται γραπτά μηνύματα (SMS).
Η μονάδα διαθέτει δύο ακιδοσειρές εκατέρωθεν, που το καθιστά ικανό να προσαρμοστεί-ενσωματωθεί ακριβώς επάνω στην πλακέτα. Με τον τρόπο αυτό καταλαμβάνει τις ψηφιακές θήρες 13 έως 2 και τις αναλογικές A0 έως A5. Επίσης καταλαμβάνει και τις εξόδους RESET, 3.3V, 5V, GND και την είσοδο Vin (input voltage). Στην πραγματικότητα όμως δεν χρησιμοποιεί, παρά μόνο τις ψηφιακές θήρες D0, D1, D7, D8 και D9. Για αυτό το λόγο στην πάνω μεριά του διαθέτει άλλες δύο σειρές από θηλυκούς συνδετήρες οι οποίοι είναι συνδεδεμένοι ένας προς ένα με κάθε ακίδα στη σειρά.

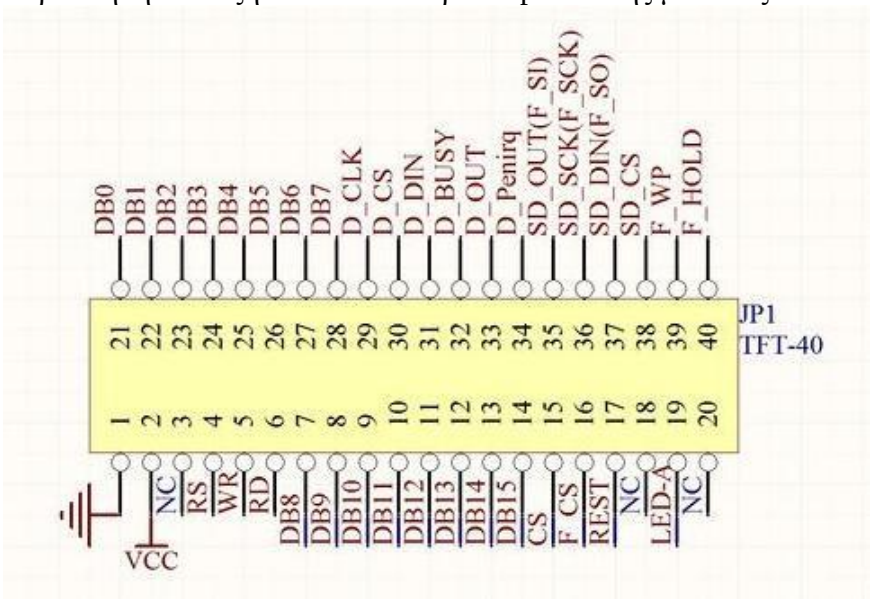


Σχήμα 5-9 Συνδεση Arduino με GPRS module

Αναλυτικότερα, η συγκεκριμένη μονάδα, σχεδιάστηκε, όπως και η βιβλιοθήκη που το συνοδεύει, για να λειτουργεί με την αναπτυξιακή πλακέτα Arduino UNO. Για αυτό το λόγο πρέπει να γίνουν κάποιες αλλαγές όσον αφορά τον τρόπο επικοινωνίας της με την πλακέτα. Η μονάδα επικοινωνεί με την πλακέτα (UNO) μέσω της σειριακής διεπαφής της, όμως με το Arduino Mega δεν έχουμε αυτήν τη δυνατότητα , πρέπει να δρομολογήσουμε την κίνηση από τη μονάδα, απευθείας στις θήρες Tx (transmit) και Rx (Receive) της πλακέτας σε χιαστή. Δηλαδή, το Tx της μονάδας, στο Rx της πλακέτας. Και το Rx της μονάδας στο Tx της πλακέτας.

5.5 Διεπαφή/Αποθήκευση

Για τη δημιουργία γραφικής απεικόνισης του μενού και των ρυθμίσεων, όπως και για τις ανάγκες αποθήκευσης δεδομένων, θα χρησιμοποιήσουμε μία περιφερειακή μονάδα η οποία ενσωματώνει μία οθόνη αφής 3,2” και sdCard (κάρτα μνήμης) module. Η μονάδα διαθέτει 40 ακροδέκτες σε παράλληλη διάταξη. Αναλυτικότερα το pinout της μονάδας:



Σχήμα 5-10 TFT pinout

Η διασύνδεση της μονάδας με την πλακέτα όσον αφορά την οθόνη και την αφή, είναι ως εξής:

Μονάδα->Πλακέτα
LEDA -> 5V
VCC -> 5V
RD -> 3.3V
GND -> GND
DB0->DB7 to pin D37->D30
DB8->DB15 to pin D22->D29
RS -> D38
WR -> D39
CS(pin15) -> D40
RSET-> D41

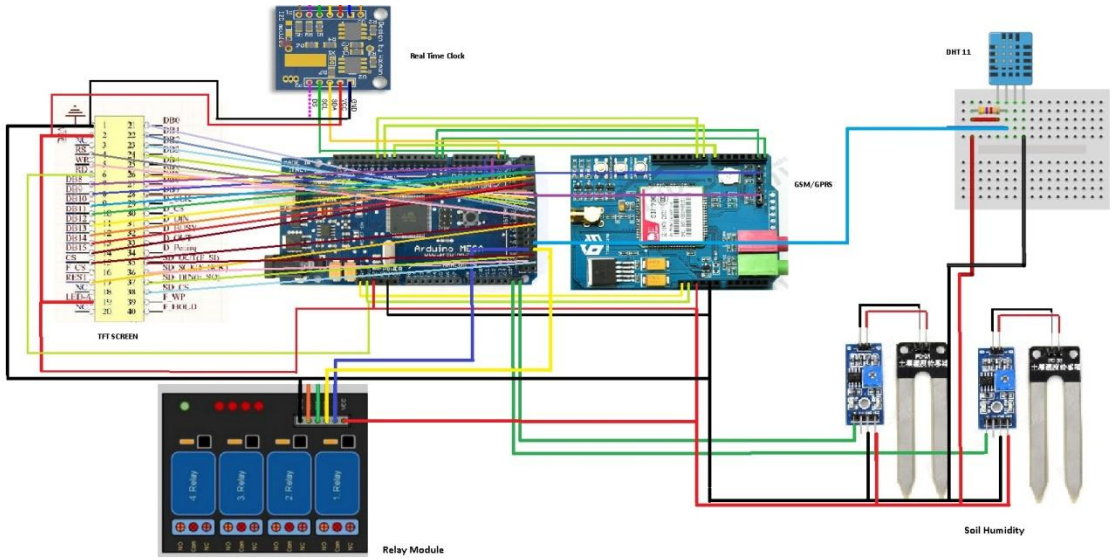
Πίνακας 5-1 TFT pinout 1

Η διασύνδεση της μονάδας όσον αφορά το SdCard Module είναι ως εξής:

Μονάδα->Πλακέτα
SCK -> D52
MISO -> D50
MOSI-> D51
CS -> D53

Πίνακας 5-2 TFT pinout 2

5.6 Ολική Συνδεσμολογία Συστήματος



Σχήμα 5-11Ολική Συνδεσμολογία Συστήματος

6. ΚΕΦΑΛΑΙΟ 6
ΥΛΟΠΟΙΗΣΗ ΛΟΓΙΣΜΙΚΟΥ

6.1 Προγραμματισμός Ελεγκτή

Μετά την επιτυχή συναρμολόγηση όλων των περιφερειακών, θα πρέπει να προγραμματιστούν και ανάλογα για να αποκτήσουν και την απαραίτητη λειτουργικότητα για την οποία επιλέχθηκαν. Για τον προγραμματισμό της αναπτυξιακής μας πλακέτας και των περιφερειακών της, χρησιμοποιήσαμε την πλατφόρμα ανάπτυξης Arduino IDE που όπως προαναφέραμε χρησιμοποιεί τον compiler της C++. Όπως και η C++, έτσι και εδώ, προγραμματίζουμε σε μία αντικειμενοστραφή γλώσσα, όπου ο σκελετός της έχει συγκεκριμένη και ιδιαίτερη δομή. Ειδικότερα, ο κώδικας χωρίζεται σε τρία κομμάτια τα οποία θα αναφερθούν με νοηματική σειρά.
Το πρώτο κομμάτι, είναι το κομμάτι όπου η αρχικοποίηση του συστήματος λαμβάνει χώρα. Είναι το πρώτο κομμάτι που εκτελεί ο compiler και είναι πιθανό το σημαντικότερο κομμάτι του κώδικα. Αυτό το κομμάτι υλοποιείται με μία «**μέθοδο**», μία συνάρτηση δηλαδή με το όνομα "**setup**", η οποία καλείται και εκτελείται από το πρόγραμμα, μόνο μία φορά.
Το δεύτερο κομμάτι είναι άλλη μία μέθοδος, την οποία το πρόγραμμα καλεί επαναλαμβανόμενα, αμέσως μετά τη μέθοδο **"setup"**. Εκεί μέσα λαμβάνει χώρα το μεγαλύτερο κομμάτι του κώδικα, το οποίο επαναλαμβάνεται συνέχεια. Το όνομα αυτής της μεθόδου, είναι **"loop" .**
Το τρίτο και πιο ενδιαφέρον κομμάτι, είναι ο «**χώρος**» έξω από τις συναρτήσεις. Είναι στην πραγματικότητα ένα υπερσύνολο του προγράμματος. Σε αυτό το «χώρο» δηλώνονται οι μεταβλητές που θέλουμε να είναι προσπελάσιμες από παντού, καλούνται και φορτώνονται οι «**Βιβλιοθήκες**» που θα χρησιμοποιήσουμε στον κώδικά μας, δηλώνονται και γράφονται οι εξωτερικές μέθοδοι οι οποίες θα μας βοηθήσουν προγραμματιστικά, και τέλος δηλώνονται και τα «**αντικείμενα**» των κλάσεων των βιβλιοθηκών.

6.1.1 Λειτουργία του Προγράμματος

Όπως αναφέραμε παραπάνω, το πρόγραμμα αφού δεσμεύσει χώρο για τις μεταβλητές , τα αντικείμενα των κλάσεων και χαρτογραφήσει και όλες τις μεθόδους, καλεί τη συνάρτηση "setup". Παρατίθεται ο κώδικας της συνάρτησης.

```
void setup()
1.  Serial.begin(57600);
2.  initGSM();
3.  initGPRS
4.  initSD();
5.  readFile("Relays.TXT");
6.  InitRelays();
7.  readFile("SCHEDUL1.TXT");
8.  InitSchedule1();
9.  readFile("SCHEDUL2.TXT");
```

```
10. InitSchedule2();
11. readFile("WATERING.TXT");
12. InitAlarms();
13. dht.begin();
14. InitRTC();
15. InitTFT();

}
```

6.1.2 Σειριακή Διεπαφή

Εδώ, το πρώτο πράγμα που κάνει η συνάρτηση, είναι να ξεκινήσει τη σειριακή διεπαφή (**Serial.begin(57600)**). Η σειριακή βιβλιοθήκη, είναι ενσωματωμένη στην πλατφόρμα και δεν χρειάζεται να την καλέσουμε ούτε να δηλώσουμε το αντικείμενο "Serial". Με τη μέθοδο "begin" και με όρισμα την ταχύτητα της επικοινωνίας(baud rate 57600), ξεκινάμε τη σειριακή σύνδεση. Μέσω της σειριακής διεπαφής θα μπορούμε να βλέπουμε τα μηνύματα του συστήματος και να αλληλεπιδρούμε με αυτό.

6.1.3 GSM/GPRS

Έπειτα, καλούνται δύο συναρτήσεις, οι οποίες αρχικοποιούν τη μονάδα GSM, συνδέεται με το δίκτυο κινητής τηλεφωνίας (**InitGSM()**) και συνδέεται και με το internet (**initGPRS()**). Αυτές οι συναρτήσεις παραπέμπουν τον compiler σε ένα άλλο κομμάτι κώδικα, όπου είναι δηλωμένες οι συναρτήσεις και έχουν και το ανάλογο περιεχόμενο. Ειδικότερα παρατίθεται ο κώδικας της συνάρτησης **InitGSM().**

```
void initGSM(){
 if (gsm.begin(9600)){
   Serial.println(F("\nGSM=READY"));
   started=true;
 }
 else Serial.println(F("\nstatus=IDLE"));
}
```

Για αυτή τη συνάρτηση, καλέσαμε τη βιβλιοθήκη **<GSM.h>** . Με τη μέθοδο "**gsm.begin**" ξεκινάμε τη σύνδεση της μονάδας με το GSM δίκτυο. Αυτή η συνάρτηση επιστρέφει **TRUE** σε περίπτωση επιτυχίας, και **FALSE** σε περίπτωση αποτυχίας. Με μία συνάρτηση **"if"** ελέγχει αυτό το αποτέλεσμα, τυπώνει το ανάλογο μήνυμα στη σειριακή θύρα, και σε περίπτωση επιτυχίας, δίνει την τιμή "**true**" στην **public Boolean** μεταβλητή "**started**".
Η επόμενη συνάρτηση που καλείται είναι η **initGPRS()**,με την οποία το σύστημα συνδέεται στο internet. Για αυτή τη συνάρτηση, χρειάστηκε να καλέσουμε τρεις βιβλιοθήκες. Την "inetGSM.h" που είναι η βιβλιοθήκη που ορίζει τις μεθόδους σύνδεσης στο internet, την "SIM900.h" που είναι η βιβλιοθήκη που χειρίζεται τις εντολές **AT** με τη βοήθεια των οποίων επικοινωνείς απευθείας με το τσιπ Sim900 και την <MemoryFree.h> με την οποία ελευθερώνουμε όση μνήμη μπορεί να ελευθερωθεί μετά από κάποια εργασία. Ακόμα δηλώσαμε και το αντικείμενο "**inet"** της πρώτης βιβλιοθήκης (InetGSM inet;).Ειδικότερα παρατίθεται ο κώδικας της συνάρτησης.

```
void initGPRS(){
❖ if(started){
    ➢ Serial.println(F("GPRS starting"));
❖ if (gprsstatus = inet.attachGPRS(GPRS_APN,GPRS_LOGIN, GPRS_PASSWORD))
    ➢ Serial.println(F("ATTACHED"));
❖ else Serial.println(("ERROR"));
❖ delay(1000);
❖ gsm.SimpleWriteln("AT+CIFSR");
❖ delay(1000);
❖ }freeMemory();

}
```

Αρχικά η συνάρτηση ελέγχει τη μεταβλητή **"started" ,** αν είναι "true",δηλαδή αν έχει συνδεθεί με επιτυχία στο GSM δίκτυο, και τότε μόνο μπαίνει και στη διαδικασία να συνεχίσει. Τυπώνει στη σειριακή θύρα ότι ξεκινάει, και κάνει προσπάθεια να συνδεθεί στο internet με τη χρήση της βιβλιοθήκης "inetGSM.h" και τη συνάρτηση **" inet.attachGPRS"** του αντικειμένου **"inet",** που παίρνει σαν ορίσματα το APN, το όνομα χρήστη και το συνθηματικό του εκάστοτε δικτύου. Ομοίως, η συνάρτηση αυτή επιστρέφει ένα Boolean αποτέλεσμα το οποίο αποθηκεύουμε στην **public boolen** μεταβλητη **gprsstatus.** Η παραπάνω μέθοδος μπαίνει σε μία **"if"** όπου ομοίως με παραπάνω, τυπώνει στη σειριακή θύρα μήνυμα ανάλογα το αποτέλεσμα. Έπειτα, μπαίνει σε αναμονή ενός δευτερολέπτου με τη χρήση της συνάρτησης "**delay"** που παίρνει ως όρισμα τα millisecond της επιθυμητής αναμονής, για να προλάβει η μονάδα να απαντήσει, αφού για να πάρει «απάντηση» από το δίκτυο χρειάζεται γύρω στο μισό δευτερόλεπτο. Όταν πάρει απάντηση, με τη συνάρτηση **"SimpleWriteln"**, γράφει εντολές **AT** στη σειριακή διεπαφή. Βάζοντας όρισμα την εντολή **"AT+CIFSR",** η μονάδα διαβάζει την διεύθυνση **IP** που της ανέθεσε το δίκτυο. Αφού μπει σε αναμονή για άλλο ένα δευτερόλεπτο, ελευθερώνει όση μνήμη μπορεί με τη συνάρτηση **FreeMemory().**

6.1.4 SdCard

Η επόμενη συνάρτηση που καλείται, είναι η **"initSD()" .** Για αυτή τη συνάρτηση καλέσαμε τρεις βιβλιοθήκες. Την "tinyFAT.h" η οποία περιέχει τις βασικές μεθόδους για το χειρισμό αρχείων αλλά και την υλοποίηση του καναλιού επικοινωνίας της πλακέτας με το SdCard module. Ακόμα, καλέσαμε την "avr/pgmspace.h" που αφορά την αποθήκευση δεδομένων στη μνήμη flash αντί της SRAM του συστήματος, και την "wire.h" η οποία ενεργοποιεί τις θήρες SDA και SCL στις οποίες συνδέεται και το SdCard module, και αναλαμβάνει τη διεκπεραίωση της μπρος και πίσω επικοινωνίας μεταξύ πλακέτας και μονάδας. Τις δύο τελευταίες βιβλιοθήκες τις χρησιμοποιεί η πρώτη. Ειδικότερα παρατίθεται ο κώδικας των συναρτήσεων.

```
void initSD(){
1) res=file.initFAT();
2) if (res!=NO_ERROR) {
```

```
        i) Serial.print(F("***** ERROR: "));
        ii) Serial.println(verboseError(res));
        iii) while (true) {};

}
}
char *verboseError(byte err){
        a. switch (err)  {
        b. case ERROR_MBR_READ_ERROR:
                i. return "Error reading MBR";
                ii. break;
        c. case ERROR_MBR_SIGNATURE:
                i. return "MBR Signature error";
                ii. break;
        d. case ERROR_MBR_INVALID_FS:
                i. return "Unsupported filesystem";
                ii. break;
        e. case ERROR_BOOTSEC_READ_ERROR:
                i. return "Error reading Boot Sector";
                ii. break;
        f. case ERROR_BOOTSEC_SIGNATURE:
                i. return "Boot Sector Signature error";
                ii. break;
        g. default:
                i. return "Unknown error";
                ii. break;

        }
}
```

Στην πραγματικότητα η αρχικοποίηση του SD module, με τη βοήθεια της βιβλιοθήκης "tinyFAT.h" δεν είναι καθόλου πολύπλοκος. Με τη μέθοδο “initFAT” του αντικειμένου “file” αρχικοποιούνται όλοι οι παράμετροι για τον χειρισμό των αρχείων σε σύστημα αρχείων “FAT16”. Η συνάρτηση επιστρέφει ένα αποτέλεσμα τύπου “**byte**” και το περνάμε στην **public byte** μεταβλητή **res.** Με τη γνωστή **“if”** ελέγχουμε το αποτέλεσμα, και με τη χρήση της συνάρτησης **char *verboseError(byte err)** τυπώνουμε το ανάλογο μήνυμα σφάλματος στη σειριακή διεπαφή. Ειδικότερα, η συνάρτηση παίρνει ως όρισμα μία μεταβλητή τύπου **byte** και συγκεκριμένα τη μεταβλητή **res,** η οποία περιέχει το αποτέλεσμα της initFAT και κάνοντας ένα σειριακό έλεγχο με τη χρήση της **switch / case** μεθόδου τυπώνει στη σειριακή θύρα το ανάλογο μήνυμα σφάλματος.
Τέλος, επιστρέφοντας στην **initSD,** εφόσον το αποτέλεσμα δεν είναι επιτυχές, θέτει το πρόγραμμα σε τέλμα, με ένα ατέρμονα βρόχο(**while(true)**).

6.2 Αρχικοποίηση Συστήματος

6.2.1 Read File

Επιστρέφοντας στη **setup** οι επόμενες γραμμές (5-12) διαβάζουν τα αρχεία με τις αποθηκευμένες ρυθμίσεις του συστήματος και τις καταχωρούν στις κατάλληλες μεταβλητές.

Αναλυτικότερα, με τη συνάρτηση **readFile()** η οποία παίρνει σαν όρισμα το όνομα του αρχείου που θέλουμε να διαβάσουμε και το καταχωρεί σε μία μεταβλητή την οποία χρησιμοποιούμε σαν buffer για να περάσουμε τις επιθυμητές τιμές στις μεταβλητές του συστήματος. Ειδικότερα παρατίθεται ο κώδικας της συνάρτησης.

```
1)  void readFile(char *filename){
2)  Serial.println(F("reading file"));
3)  if (file.exists(filename))  {
4)  res=file.openFile(filename, FILEMODE_TEXT_READ);
5)  if (res==NO_ERROR)   {
    a)  result=0;
    b)  result=file.readLn(textBuffer, 5);


    c)  Serial.println(result);
    d)  if (result!=FILE_IS_EMPTY)      {
        i)   if (result==BUFFER_OVERFLOW){
        ii)  Serial.print(textBuffer);
        iii) }
        iv)  else{
        v)   Serial.println(textBuffer);
        vi)  }
    e)  }
    f)  else
        i)   Serial.println(F("** ERROR: File is empty..."));
    g)  Serial.println();
    h)  file.closeFile();
6)  }
7)  else   {
    a)  switch(res){
    b)  case ERROR_ANOTHER_FILE_OPEN:
        i)   Serial.println(F("** ERROR: Another file is already open..."));
        ii)  break;
    c)  default:
        i)   Serial.print(F("** ERROR: "));
        ii)  Serial.println(res, HEX);
        iii) break;
    d)  }
8)  }
9)  }
10) else
11) Serial.println("** ERROR: 'TEXTFILE.TXT' does not exist...");


12) Serial.println();
13) Serial.println();
14) Serial.println(F("***** All done... *****"));
15) freeMemory();
16) }
```

Η συνάρτηση, με τη μέθοδο "**exixts"** του αντικειμένου **"file"** με όρισμα το όνομα του αρχείου, ελέγχει αν υπάρχει το αρχείο που ζητάμε, και αν υπάρχει, με τη μέθοδο **readLn** διαβάζει τους πέντε πρώτους χαρακτήρες του αρχείου και τους καταχωρεί στην **public char textbuffer[5]**

μεταβλητή. Η μεταβλητή και το μέγεθος του buffer έχουν περαστεί ως ορίσματα στη μέθοδο **readLn.** Τα αποτελέσματα που επιστρέφουν οι **openFile** και **readLn** καταχωρούνται στις μεταβλητές **res** και **result** αντίστοιχα. Έπειτα, με τη χρήση των **"if"** και **"switch / case"** μεθόδων, γίνεται έλεγχος και τυπώνεται το ανάλογο μήνυμα σφάλματος στη σειριακή θύρα. Τέλος, με τη συνάρτηση **FreeMemory()** ελευθερώνεται η καταληφθείσα μνήμη.
Αμέσως μετά, εκτελείται η συνάρτηση **initRelays()** η οποία μετατρέπει το περιεχόμενο της μεταβλητής **textBuffer**(πίνακας χαρακτήρων) σε ακέραιο **(int)** με τη χρήση της μεθόδου **toInt()** του αντικειμένου **String** και το καταχωρεί στην **public int relays** μεταβλητή. Ειδικότερα παρατίθεται ο κώδικας της συνάρτησης.

```
void InitRelays(){
  relays=String(textBuffer[0]).toInt();
}
```

6.2.2 InitSchedule

Η παραπάνω διαδικασία, εκτελείται άλλες τρεις φορές διαβάζοντας τα αρχεία "SCHEDUL1.TXT", "SCHEDUL2.TXT" και "WATERING.TXT" και εκτελώντας τις συναρτήσεις **InitSchedule1(), InitSchedule2()** και **InitAlarms().** Ειδικότερα παρατίθενται οι κώδικες των συναρτήσεων.

```
void InitSchedule1(){
        o  Serial.println(F("Init schedules 1"));
        o  Serial.println(textBuffer);
        o  String hh=String(textBuffer[0]);
        o  h1=hh.toInt();
        o  String mm=String(textBuffer[2]);
        o  m1=mm.toInt();

}
void InitSchedule2(){
        o  Serial.println(F("Init schedules 2"));
        o  Serial.println(textBuffer);
        o  String hh=String(textBuffer[0]);
        o  h2=hh.toInt();
        o  String mm=String(textBuffer[2]);
        o  m2=mm.toInt();

}
```

Οι δύο αυτές συναρτήσεις, διαβάζουν τα δύο αρχεία με τα αποθηκευμένα προγράμματα ποτίσματος και μετατρέποντας σε ακέραιους τα περιεχόμενα της μεταβλητής **textBuffer,** τα καταχωρούν στις μεταβλητές που χρησιμοποιεί το πρόγραμμα.
Έπειτα, αφού διαβάσει το αρχείο "WATERING.TXT" που περιέχει των αριθμό των ποτισμάτων που επιθυμούμε να γίνονται ημερησίως, εκτελείται η συνάρτηση **InitAlarms()** η οποία ανάλογα τον αριθμό των ποτισμάτων που έχουμε ορίσει, καταχωρεί τα προγράμματα του ποτίσματος και μετά καταχωρεί και τις υπόλοιπες περιοδικές εργασίες του συστήματος. Ειδικότερα παρατίθεται ο κώδικας της συνάρτησης

```
1) void InitAlarms(){////////////////////////////////////////////////////////////////Initialize the alarms
2) if(textBuffer[0]=='1'){
   a) Serial.println(F("1 watering has been chosen"));
   b) Alarm.alarmRepeat(h1,m1,0,Potise);  ////////////////////////////////  MORNING  WATERING
3) }else if(textBuffer[2]=='2'){
   a) Serial.println(F("2 waterings have been chosen"));
   b) Alarm.alarmRepeat(h1,m1,0,Potise);  ////////////////////////////////  MORNING  WATERING
   c) Alarm.alarmRepeat(h2,m2,0,Potise);  ////////////////////////////////  NIGHT  WATERING
4) }
5) Alarm.timerRepeat(60,readIfsms);  ////////////////////////////////  Read  SMS
6) Alarm.timerRepeat(180,SqlPost);
7) }
```

Για αυτή τη συνάρτηση, καλέσαμε τρεις βιβλιοθήκες. Την "Time.h" η οποία περιλαμβάνει τις μεθόδους για το χειρισμό της ώρας συστήματος, την "TimeAlarms.h" η οποία περιέχει τις μεθόδους με τις οποίες ορίζουμε την περιοδικότητα εκτέλεσης εργασιών στο σύστημα. Αυτό επιτυγχάνεται με τη χρήση **watchdog** και **interrupt.** Το **interrupt** είναι μία λειτουργία του μικροελεγκτή, η οποία ενεργοποιείται με κάποια ενέργεια, η οποία επιτρέπει στο πρόγραμμα να διακόψει τη κανονική του ροή του και να εκτελέσει μία λειτουργία, εν προκειμένω μία συνάρτηση. Το **watchdog** είναι άλλη μία λειτουργία του μικροελεγκτή, η οποία δημιουργεί ένα συμβάν σε προγραμματισμένο χρόνο, το οποίο συμβάν στην περίπτωση μας είναι αυτό που ενεργοποιεί το **interrupt.** Η βιβλιοθήκη που υλοποιεί το **watchdog,** είναι η "avr/wdt.h". Η βιβλιοθήκη "TimeAlarms.h", χρησιμοποιεί τις άλλες δυο.
Σε αυτή τη συνάρτηση, θα χρησιμοποιήσουμε μόνο δύο από τις μεθόδους της βιβλιοθήκης "TimeAlarms.h". Την **timerRepeat(milliseconds, function),** η οποία ορίζει το χρόνο στον οποίο θα επαναλαμβάνεται η επιθυμητή συνάρτηση. Ο χρόνος και η συνάρτηση, περνούν ως ορίσματα στη μέθοδο, και την **alarmRepeat(hour, minute, second, function),** η οποία ορίζει την ακριβή ώρα που θα εκτελείται η επιθυμητή συνάρτηση ημερησίως. Η ώρα και η συνάρτηση περνούν ως ορίσματα στη μέθοδο.

6.2.3 Πότισμα

Αναλυτικότερα, στις γραμμές 2.b , 3.b και 3.c, προγραμματίζουμε σε διαφορετικές ώρες, την εκτέλεση της συνάρτησης **Potise().** Η συνάρτηση αυτή, ελέγχει αν το έδαφος είναι στεγνό αρκετά και στου δύο χώρους και αν ναι ανοίγει τα ανάλογα ρελέ και μετά από δέκα λεπτά, τα κλείνει. Ειδικότερα παρατίθεται ο κώδικας της συνάρτησης.

```
1) void Potise(){
2) if(Soil<80){
   a) relay.on(1);
   b) Status="Live";
   c) Alarm.timerOnce(600, KleiseVanes(1));
3) }else{
   a) sms.SendSMS(PhoneNumber,"Watering Cancelled.Soil too wet to water on Area 1...");
4) }
5) if(Soil2<80){
   a) relay.on(2);
   b) Status="Live";
```

```
    c) Alarm.timerOnce(600, KleiseVanes(2));
6) }else{
    a) sms.SendSMS(PhoneNumber, "Watering Cancelled.Soil too wet to water on Area 2...");
7) }
8) }
```

Στη συνάρτηση αυτή εμπλέκονται άλλες δύο βιβλιοθήκες. Η **"relay8.h"** η οποία περιέχει τις μεθόδους χειρισμού της πλακέτας των ρελέ, και η **"sms.h"** η οποία περιέχει τις μεθόδους για την αποστολή και ανάγνωση γραπτών μηνυμάτων. Αναλυτικότερα, στις γραμμές 2 και 5, η συνάρτηση ελέγχει τις 2 μεταβλητές **"Soil"** και **"Soil2"** οι οποίες περιέχουν την τιμή μέτρησης επί τις εκατό(%) της υγρασίας του χώματος του κάθε χώρου. Αν η υγρασία ξεπερνά το 80%, τότε το σύστημα δεν ποτίζει .Στις γραμμές 2.a και 5.a, χρησιμοποιούμε το αντικείμενο "**relay"** της κλάσης "**relay8"** που παίρνει ως ορίσματα, τις ψηφιακές θήρες που είναι συνδεδεμένο το κάθε ρελέ**("relay8 relay(48,49)").** Εδώ χρησιμοποιούμε τη μέθοδο **on(relay number)** του αντικειμένου **relay,** η οποία αλλάζει την κατάσταση του επιθυμητού ρελέ από NO(**normal open**) σε NC(**normal closed**). Ο αριθμός του ρελέ, περνά σαν όρισμα στη μέθοδο. Στις γραμμές 2.b και 5.b, ορίζει την κατάσταση των δύο ρελέ ως **"Live"** ότι δηλαδή το σύστημα αυτή τη στιγμή ποτίζει στο συγκεκριμένο χώρο καταχωρώντας την τιμή αυτή στις **public String Status** και **Status2** μεταβλητές. Στις γραμμές 2.c και 5.c, προγραμματίζει τη συνάρτηση "**KleiseVanes()"** να εκτελεστεί μόνο μία φορά, σε 10 λεπτά με τη χρήση της μεθόδου "**timerOnce"** της κλάσης **"Alarm".**
Η συνάρτηση "**KleiseVanes(),** με όρισμα τον αριθμό του ρελέ, αλλάζει την κατάσταση του επιθυμητού ρελέ από NC σε NO κλείνοντας τις ανάλογες ηλεκτροβάνες. Έπειτα, ορίζει την κατάσταση του ανάλογου χώρου, σε **"Idle"** καταχωρώντας την τιμή αυτή στην εκάστοτε μεταβλητή, "**Status"** ή "**Status2".** Ειδικότερα παρατίθεται ο κώδικας της συνάρτησης.

```
void KleiseVanes(int r){
  relay.off(r);
  if(r==1){
     Status="Idle";
  }else if(r==2){
    Status2="Idle";
  }
 }
```

Τέλος, στις γραμμές 3.a και 6.a, αν ο έλεγχος δεν κρίνει ότι το σύστημα πρέπει να ποτίσει, στέλνει ένα γραπτό μήνυμα στον χρήστη για να τον ενημερώσει ότι δεν πότισε, με τη χρήση της μεθόδου **"SendSMS(number, text)** της κλάσης **"sms".** Ο αριθμός τηλεφώνου και το κείμενο του μηνύματος, περνούν ως ορίσματα στη μέθοδο.

6.2.4 Έλεγχος SMS

Στη γραμμή 5 της συνάρτησης **"InitAlarms",** προγραμματίζουμε την περιοδική εκτέλεση της συνάρτησης **"readIfsms()"** κάθε ένα λεπτό, η οποία ελέγχει αν υπάρχει γραπτό μήνυμα και αν αυτό το μήνυμα περιέχει κάποια εντολή ελέγχου, να πράξει την ανάλογη ενέργεια. Τέλος, να διαγράψει όλα τα μηνύματα από τη μνήμη. Ειδικότερα παρατίθεται ο κώδικας της συνάρτησης.

```
1) void readIfsms(){
2) Serial.println(F("Reading SMS..."));
```

```
3) int pos=0;
4) if(started){
   a) pos=sms.IsSMSPresent(SMS_ALL);
   b) if(pos){
   c) Serial.println(F("Receiving SMS"));
   d) Serial.println(pos);
   e) sms.GetSMS(pos,n,smsbuffer,100);
      i) Serial.println(n);
      ii) Serial.println(smsbuffer);
      iii) CheckSMS();
      iv) delsms();
   f) }
5) }
6) }
```

Στη γραμμή 3, δηλώνουμε τη μεταβλητή **"pos"** η οποία θα αντιπροσωπεύει τη θέση του γραπτού μηνύματος στη μνήμη της GSM μονάδας. Έπειτα, στη γραμμή 3, αφού ελέγξει πάλι ότι η μονάδα είναι συνδεδεμένη με το δίκτυο GSM, τότε με τη χρήση της μεθόδου **"IsSMSPresent()"** με όρισμα τον τύπο του SMS στη γραμμή 4.a, ελέγχει αν υπάρχουν αποθηκευμένα γραπτά μηνύματα, αναγνωσμένα ή μη, και επιστρέφει τη θέση στην οποία είναι αποθηκευμένα. Αφού ελέγξει ότι υπάρχει μήνυμα, τότε με τη μέθοδο **"GetSMS()"** στη γραμμή 4.e, ανοίγει το μήνυμα στην θέση που υποδεικνύει η **"pos",** το αποθηκεύει στη **public String smsbuffer** μεταβλητή με μέγεθος buffer 100 χαρακτήρων. Η θέση, το όνομα της μεταβλητής που θα χρησιμοποιηθεί ως buffer, όπως και το μέγεθος του buffer, περνούν ως όρισμα στη μέθοδο. Στη συνέχεια, στη γραμμή 4.e.iii γίνεται έλεγχος του περιεχομένου του μηνύματος. Αν αυτό περιέχει κάποια εντολή ελέγχου, προβαίνει στην ανάλογη ενέργεια. Ειδικότερα παρατίθεται ο κώδικας της συνάρτησης.

```
1) void CheckSMS(){
2) if(!strcmp(smsbuffer,"R1ON")){
3) relay.on(1);
4) Status="live";
5) }
6) else if(!strcmp(smsbuffer,"R2ON")){
7) relay.on(2);
8) Status2="live";
9) }
10) else if(!strcmp(smsbuffer,"ALLON")){
   a) relay.allOn();
   b) Status="Live";
   c) Status2="live";
11) }////////////////////////////////////////////////////////////////////////////////
12) else if(!strcmp(smsbuffer,"R1OFF")){
   a) relay.off(1);
   b) Status="Idle";
13) }
14) else if(!strcmp(smsbuffer,"R2OFF")){
   a) relay.off(2);
   b) Status2="Idle";
15) }
16) else if(!strcmp(smsbuffer,"ALLOFF")){
   a) relay.allOff();
```

```
    b) Status="Idle";
    c) Status2="Idle";
17) }
18) else
    a) Serial.println(F("prama"));
19) }
```

Εδώ, η συνάρτηση ελέγχει αν το κείμενο του μηνύματος περιέχει μία από τις ακόλουθες εντολές, με τη χρήση της εντολής "strcmp() η οποία συγκρίνει δύο String μεταξύ τους και επιστρέφει "true" αν τα δύο String είναι ίδια και "false" αν όχι, "R1ON και R2ON, που θέτουν τα ρελέ 1 και 2 στην κατάσταση "ON" με τη χρήση της μεθόδου **"on()"** του αντικειμένου **"relay"** με όρισμα τον αριθμό του ρελέ, τις R1OFF και R2OFF, που θέτουν τα ρελέ 1 και 2 στην κατάσταση "OFF"με τη χρήση της μεθόδου **"off"** του αντικειμένου **"relay"** με όρισμα τον αριθμό του ρελέ, την "ALLON" που θέτει όλα τα ρελέ στην κατάσταση "ON" με τη χρήση της μεθόδου **"allOn()"** του αντικειμένου **"relay",** και την "ALLOFF" που θέτει όλα τα ρελέ στην κατάσταση "OFF"με τη χρήση της μεθόδου **"allOff()"** του αντικειμένου **"relay".** Σε κάθε περίπτωση, καταχωρεί την κατάσταση(**"Live"/"Idle"**) του κάθε ρελέ στις ανάλογες μεταβλητές **("Status"/"Status2").**
Στη συνέχεια, στη γραμμή 4.e.iv της συνάρτησης **"readIfsms()"**, καλείται η συνάρτηση **"delsms()",** η οποία διαγράφει όλα τα μηνύματα από τη μνήμη. Ειδικότερα παρατίθεται ο κώδικας της συνάρτησης.

```
1)  void delsms(){
2)  Serial.println(F("Checking SIM card for new messages..."));
3)  for (int i=0; i<10; i++){
    a) int pos=sms.IsSMSPresent(SMS_ALL);
    b) if (pos!=0){
    c) Serial.print(F("\nFound SMS at the position "));
    d) Serial.println(pos);
    e) if (sms.DeleteSMS(pos)==1){
        i) Serial.print(F("\nDeleted SMS at the position "));
        ii) Serial.println(pos);
    f) }
    g) else
    h) {
        i) Serial.print(F("\nCant delete SMS at the position "));
        ii) Serial.println(pos);
    i) }
    j) }
4)  }
5)  }
```

Αναλυτικά, η συνάρτηση στη γραμμή 3.a ελέγχει αν υπάρχει αποθηκευμένο μήνυμα με τη χρήση της μεθόδου **"IsSMSPresent()"** αναγνωσμένο ή μη, και στη συνέχεια στη γραμμή 3.e το διαγράφει, με τη χρήση της μεθόδου **"DeleteSMS(position)"** με όρισμα τη θέση του μηνύματος. Ελέγχει αν η ενέργεια ήταν επιτυχής ή όχι, και τυπώνει το ανάλογο μήνυμα στη σειριακή θύρα. Αυτό θα επαναληφτεί δέκα φορές, με τη χρήση της **"for"** στη γραμμή 3.

6.2.5 SQL Post

Στη γραμμή 6 της **"InitAlarms()"** ορίζεται η περιοδική εκτέλεση της συνάρτησης **"SqlPost()"** με περιοδικότητα, τρία λεπτά. Η συνάρτηση αυτή εκτελεί ένα **php post** σε ένα απομακρυσμένο Apache server όπου καταγράφει τα δεδομένα του περιβάλλοντος και την κατάσταση του συστήματος. Ειδικότερα παρατίθεται ο κώδικας της συνάρτησης.

```
1)  void  SqlPost(){
2)  String str1;
3)  char str2[120];
4)  if(gprsstatus){
        i)   Serial.println(F("Posting to SqlServer..."));
             (1)  char temp[5];
             (2)  char humid[5];
             (3)  char soil[5];
             (4)  char statu[6];
                  (i)  str1="Temp=" + String(int(Temp)) + "&Humid=" + String(int(Humid)) + "&Soil=" +
                       String(Soil)+ "&Soil2" + String(Soil2) + "&Status" + String(Status) + "&Status2" +
                       String(Status2);
             (5)  str1.toCharArray(str2,120);
        ii)  numdata1=inet.httpPOST("http://nuovo.gr", 80, "giannis/arduino/sensors.php", str2 ,msg1, 50);
        iii) delay(3000);
        iv)  freeMemory();
5)  }
6)  else
7)  Serial.println(F("GPRS is NOT attached!"));
8)  }
```

Εδώ η συνάρτηση επικοινωνεί με τον απομακρυσμένο Server, και μέσω του Apache καταγράφει τα δεδομένα στον MySQL, με τη χρήση ενός αιτήματος PHP. Το αίτημα πρέπει να έχει τη μορφή **"http://ServerAdress:port/PhpScriptFolder/PhpScript.php?value1=xx&value2=xx&value3=xx" κλπ.**
Από τη μεριά του server θα πρέπει να υπάρχει ένα **"PhpScript.php"** το οποίο θα απαντήσει στο αίτημα, θα αποθηκεύσει τις τιμές των μεταβλητών με τη χρήση της εντολής "**$_GET"** της **PHP,** και θα τις εισάγει στη βάση δεδομένων.
Για αυτή τη συνάρτηση θα χρησιμοποιήσουμε τη βιβλιοθήκη "inetGSM.h" και τη μέθοδο "httpPOST(**ServerAddress,port,ScriptLocation,Data,answer,BufferSize**)" του αντικειμένου **"inet"** στη γραμμή 5.ii, με ορίσματα τη διεύθυνση του server, την θύρα που "ακούει" ο server, τα δεδομένα σε μορφή **"?value1=xx&value2=xx&value3=xx"** , το όνομα της μεταβλητής που θα αποθηκευθεί η απάντηση από το server (**http response**) και το μέγεθος του buffer αντίστοιχα.Τα δεδομένα, τα αποθηκεύουμε στην **Public char Str2[]** μεταβλητή στις γραμμές 4.i και 5 κολλώντας μία-μία τις τιμές στην **Public String str1** και μετατρέποντας την σε πίνακα χαρακτήρων με τη μέθοδο "**toCharArray()**" της κλάσης **"String"** που παίρνει ως όρισμα το όνομα της μεταβλητής που θα αποθηκευτεί το αποτέλεσμα και το μέγεθος του buffer. Όλα τα παραπάνω θα εκτελεστούν μόνο αν η μονάδα έχει συνδεθεί στο internet, και αυτό θα επιτευχθεί με τον έλεγχο της μεταβλητής "**gprsstatus"** με τη χρήση της **"if"** στη γραμμή 4, η οποία παίρνει τιμή "true" αν η σύνδεση ήταν επιτυχής και "false"αν όχι. Αν το σύστημα είναι συνδεδεμένο με το internet, τότε θα εκτελέσει την "**httpPOST**", θα βάλει το σύστημα σε αναμονή για τρία δευτερόλεπτα με τη χρήση της **"delay()"** στη γραμμή 5.iii και θα ελευθερώσει όση μνήμη μπορεί. Αλλιώς θα τυπώσει στη σειριακή θύρα το ανάλογο μήνυμα.

6.2.6 Θερμοκρασία/Υγρασία Περιβάλλοντος

Στη γραμμή 13 της **"setup"** εκτελείται η μέθοδος **"begin()"** του αντικειμένου **"dht"** της κλάσης **"DHT".** Στην δήλωση του αντικειμένου **"dht" (DHT dht(DHTPIN, DHTTYPE))**περνάμε ως ορίσματα, την ψηφιακή θύρα που είναι συνδεδεμένος ο αισθητήρας και τον τύπο του τσιπ που έχουμε. Εν προκειμένω το **"DHT11".** Για την έναρξη της λειτουργίας του αισθητήρα χρησιμοποιούμε τη βιβλιοθήκη "DHT.h".

Ώρα Συστήματος

Στη γραμμή 14 της **"setup",** εκτελείται η συνάρτηση **"InitRTC().** Η συνάρτηση αυτή, αρχικοποιεί και ξεκινά τη μονάδα **RTC** και συγχρονίζει την ώρα συστήματος με την ώρα από τη μονάδα. Ειδικότερα παρατίθεται ο κώδικας της συνάρτησης.

```
1) void InitRTC(){
2) Serial.println(F("Initialising RTC..."));
3) rtc.begin();
4) if ( !rtc.isrunning()) {
        (1) Serial.println(F("RTC IS NOT running!"));
        (2) rtc.adjust(DateTime(__DATE__, __TIME__));
5) }else{
   a) Serial.println(F(" RTC is running....time will be set according to rtc..."));
   b) DateTime now1 =rtc.now();
   c) }
6) setTime(now1.hour(),now1.minute(),now1.second(),now1.day(),now1.month(),now1.year());
7) }
```

Αναλυτικά στη γραμμή 3, χρησιμοποιούμε τη μέθοδο **"begin()"** του αντικειμένου **"rtc"** της κλάσης "**RTC_DS1307".** Με αυτή τη μέθοδο αρχικοποιούμε και ξεκινάμε τη μονάδα **"RTC".** Έπειτα, ελέγχουμε αν λειτουργεί η μονάδα, με τη χρήση της μεθόδου **"isrunning()"** στη γραμμή 4, η οποία επιστρέφει **"true"** αν ναι και **"false"** αν όχι. Στην περίπτωση που δεν λειτουργεί, στη γραμμή 4.2 ρυθμίζουμε τις τιμές της κλάσης "**RTC_DS1307"** με τις σταθερές __DATE__ και __TIME__, τις οποίες ορίζει ο compiler με τις τιμές της ώρας και της ημερομηνίας, της στιγμής που γίνεται το compiling του κώδικα. Στην περίπτωση που λειτουργεί, απλά συγχρονίζουμε το σύστημα με τη μονάδα **RTC.** Αρχικά ορίζουμε μία μεταβλητή **"now1"** τύπου "**DateTime"** που μπορεί να αποθηκεύσει την ώρα από τη μονάδα **RTC** την οποία παίρνουμε με τη μέθοδο **"now()"** στη γραμμή 5.b. Τέλος, ρυθμίζουμε την ώρα και την ημερομηνία του συστήματος με τη μέθοδο "**setTime()"** από τη βιβλιοθήκη "Time.h" στη γραμμή 6.

6.2.7 Οθόνη αφής

Στην τελευταία γραμμή της **"setup"** εκτελείται η συνάρτηση **"initTFT()"** η οποία αρχικοποιεί τις παραμέτρους που χρειάζεται για να λειτουργήσει η οθόνη αφής. Ειδικότερα παρατίθεται ο κώδικας της συνάρτησης.

```
1) void InitTFT(){
2) myGLCD.InitLCD();
```

```
3)  myGLCD.clrScr();
4)  myGLCD.setFont(SmallFont);
5)  myTouch.InitTouch();
6)  myTouch.setPrecision(PREC_MEDIUM);
7)  myButtons.setTextFont(BigFont);
8)  myButtons.setSymbolFont(Dingbats1_XL);
9)  myButtons.setButtonColors(VGA_WHITE, VGA_GRAY, VGA_WHITE, VGA_RED, VGA_BLUE);
10) }
```

Για αυτή τη συνάρτηση, θα χρειαστούμε τρεις βιβλιοθήκες. Την "UTFT.h", η οποία περιέχει τις μεθόδους για την έναρξη και παραμετροποίηση της γραφικής απεικόνισης της οθόνης, την "UTouch.h" που περιέχει τις μεθόδους για την παραμετροποίηση και υλοποίηση της λειτουργίας αφής και την "UTFT_Buttons.h" που περιέχει τις μεθόδους για την παραμετροποίηση, υλοποίηση και χειρισμό των κουμπιών. Για το σκοπό αυτό δηλώνονται και τρία αντικείμενα των κλάσεων **"UTFT", "UTouch"** και **"UTFTButtons".**Το **"myGLCD"** **(UTFT myGLCD(ITDB32S,38,39,40,41))** που παίρνει ως ορίσματα το τσιπ που χρησιμοποιεί η συγκεκριμένη οθόνη και τις ψηφιακές θήρες που χρησιμοποιώ για την οθόνη. Το **"myTouch"** **(UTouch myTouch(42,43,44,45,46))**που παίρνει ως ορίσματα τις ψηφιακές θήρες που χρησιμοποιώ για το touch και τέλος το **"myButtons"(UTFT_Buttons myButtons(&myGLCD, &myTouch))** που παίρνει ως ορίσματα τα δύο παραπάνω αντικείμενα. Αναλυτικά στις γραμμές 2 και 3 με τη χρήση της μεθόδου **"InitTFT()"** του αντικειμένου **"myGLCD"** αρχικοποιούμε την οθόνη και με τη χρήση της μεθόδου **"clrScr()"** του ίδιου αντικειμένου, καθαρίζει την οθόνη από πιθανά προηγούμενα κατάλοιπα στο buffer της. Στη γραμμή 4, με τη μέθοδο **"setFont()"** ορίζουμε τι γραμματοσειρά θα χρησιμοποιήσουμε. Σαν όρισμα στη μέθοδο περνάμε την δεσμευμένη από τη βιβλιοθήκη μεταβλητή **"SmallFont"** η οποία μας καθορίζει τη γραμματοσειρά. Στις γραμμές 5 και 6 με τις μεθόδους **"InitTouch()"** και **"setPrecision()"** αρχικοποιούμε τη λειτουργία της αφής και ορίζουμε την ευαισθησία της αντίστοιχα. Στις γραμμές 7,8 και 9, με τις μεθόδους **"setTextFont()", "setSymbolFont()"** και **"setButtonColors()",** ορίζουμε τη γραμματοσειρά που θα χρησιμοποιήσουμε στα κουμπιά, τα σύμβολα που θα χρησιμοποιήσουμε , και τα χρώματα που θα έχουν.

Όταν το πρόγραμμα τελειώσει με την αρχικοποίηση του συστήματος, φύγει από τη **"setup()"** δηλαδή, πηγαίνει αμέσως στη μέθοδο **"loop()"** την οποία και θα εκτελεί συνεχώς. Σε αυτή τη μέθοδο μέσα, θα υλοποιηθεί και η γραφική απεικόνιση του μενού μας.

6.3 Μενού

Για τη γραφική απεικόνιση του μενού μας, χρησιμοποιούμε μια tft touch οθόνη, η οποία ενεργοποιείται, όπως αναφέρεται και παραπάνω, με τις βιβλιοθήκες "UTFT.h", "UTouch.h" και "UButtons.h". Για να μπορέσουμε αρχικά να απεικονίσουμε κάτι στην οθόνη και να είναι λειτουργικά ορατό, πχ ένα κείμενο, θα πρέπει να τοποθετήσουμε το αντικείμενο σε ένα ατέρμονα βρόχο, λόγω της συνεχούς ανανέωσης (refresh) της οθόνης. Αλλιώς, θα εμφανιστεί για κάποια εκατοστά του δευτερολέπτου, και θα εξαφανιστεί. Για αυτό το λόγο πρέπει να δημιουργήσουμε μία συνάρτηση, η οποία θα απεικονίζει την εκάστοτε οθόνη με τα αντικείμενα της. Το μενού θα έχει μία δενδροειδή μορφή οπότε από τη μέθοδο **"loop()"**, θα παραπεμφθούμε

στην πρώτη συνάρτηση του μενού, την **MainScreen().** Τα μοναδικά αντικείμενα που θα χρησιμοποιήσουμε, είναι κουμπιά και κείμενο. Παρατίθεται ο κώδικας της **"loop()".**

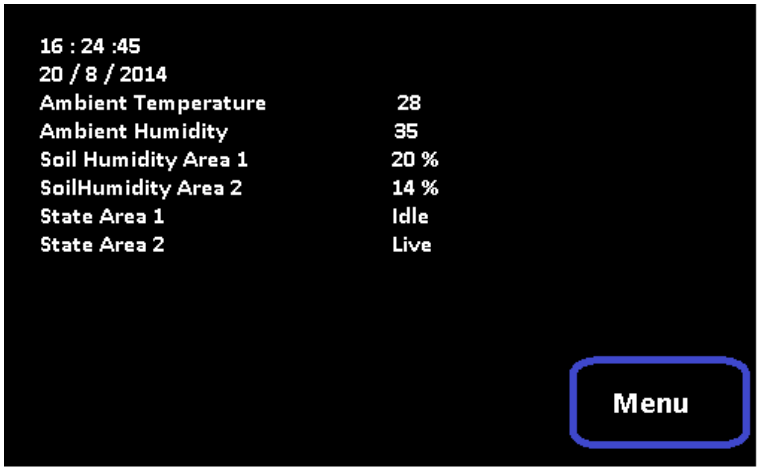
```
void loop(){
        MainScreen();
}
```

Με τη σειρά της η **MainScreen()** είναι μια πολύ ενδιαφέρουσα συνάρτηση. Αρχικά, δηλώνονται τα κουμπιά σαν μεταβλητές, καταχωρούνται τα γραφικά αντικείμενα(κουμπιά) στις μεταβλητές τους, και έπειτα λέμε στο αντικείμενο **myButtons** να μας τα ζωγραφίσει. Να σημειωθεί ότι τα κουμπιά σαν αντικείμενα, αντίθετα με τα υπόλοιπα της βιβλιοθήκης, δεν χρειάζεται να εισαχθούν σε βρόχο για την απεικόνιση τους. Έπειτα, περνάμε στο βρόχο(**while(1)**) όπου τυπώνονται τα κείμενα και υλοποιούνται οι οι **"Action Listeners"** των κουμπιών. Ειδικότερα παρατίθεται ο κώδικας της **MainScreen().**

```
1)  void MainScreen(){
2)  int MenuButton,pressed_button=NULL;
3)  boolean default_colors = true;
4)  MenuButton=myButtons.addButton(200,199,100,40,"Menu");
5)  //freeMemory();
6)  myGLCD.clrScr();
7)  myButtons.drawButton(MenuButton);
8)  while(1){
9)  DateTime d = rtc.now();
10) Alarm.delay(5);
11) Serial.println(freeMemory());
12) myGLCD.print(String(d.hour()),10,10);
13) myGLCD.print(":",30,10);
14) myGLCD.print(String(d.minute()),50,10);
15) myGLCD.print(":",70,10);
16) myGLCD.print(String(d.second()),90,10);
17) myGLCD.print(String(d.day()),10,30);
18) myGLCD.print("/",30,30);
19) myGLCD.print(String(d.month()),50,30);
20) myGLCD.print("/",70,30);
21) myGLCD.print(String(d.year()),90,30);
22) myGLCD.print("Ambient Temperature:",10,50);
23) myGLCD.printNumF(Temp,2,200,50);
24) myGLCD.print("Ambient Humidity",10,70);
25) myGLCD.printNumF(Humid,2,200,70);
26) myGLCD.print("Soil Humidity Area 1",10,90);
27) myGLCD.printNumF(Soil,2,200,90);
28) myGLCD.print("%",2,200,90);
29) myGLCD.print("Soil Humidity Area 2",10,110);
30) myGLCD.printNumF(Soil2,2,200,110);
31) myGLCD.print("%",2,200,110);
32) myGLCD.print("Area1 State",10,130);
33) myGLCD.print(Status,200,130);
34) myGLCD.print("Area2 State",10,150);
35) myGLCD.print(Status2,200,150);
36) if (myTouch.dataAvailable() == true)
37) {
    a) pressed_button = myButtons.checkButtons();
    b) if(pressed_button==MenuButton){
    c) myButtons.deleteAllButtons();
```

```
    d) MainMenu();
    e) }
38) }
39) readDHT();
40) ReadSoil();
41) }
42) }
```

Στη γραμμή 4, καταχωρώ το αντικείμενο(κουμπί) στη μεταβλητή του με τη μέθοδο **addButton(xAxis,yAxis,width,Height)** του αντικειμένου **myButtons** και τέλος στη γραμμή 7, με τη μέθοδο **drtawButton(ButtonName)** εμφανίζω το κουμπί στην επιθυμητή θέση με το επιθυμητό μέγεθος. Λίγο πριν εισέλθουμε στη **while(1),** εκτελείται η μέθοδος **clrScr()** του αντικειμένου **myGLCD,** ούτως ώστε κάθε φορά που εκτελείται από την αρχή η συνάρτηση, είτε γυρίζοντας πίσω από άλλο μενού είτε καλούμενη από τη **loop(),** η συνάρτηση θα καθαρίζει το buffer της οθόνης για να μην αφήνει γραφικά κατάλοιπα. Μπαίνοντας στη **while(1),** επειδή **η** πρώτη οθόνη θα απεικονίζει την ώρα και τις συνθήκες του περιβάλλοντος, αρχικά δηλώνουμε μια μεταβλητή του τύπου **DateTime** και της καταχωρούμε κάθε φορά την ώρα του συστήματος. Έπειτα με τη χρήση της μεθόδου **print(text,xAxis,yAxis)** τυπώνουμε το κείμενο στο επιθυμητό σημείο. Το μέγεθος το έχουμε επιλέξει νωρίτερα στη **setup().**
Περεταίρω, προχωράμε στο χειρισμό των συμβάντων αφής. Με τη μέθοδο **dataAvailable()** του αντικειμένου **myTouch** ελέγχουμε αν έχει υπάρξει κάποιο συμβάν. Η μέθοδος επιστρέφει **true** ή **false.** Κάθε κουμπί κατά τη δημιουργία του, επιστρέφει ένα μοναδικό ακέραιο αριθμό. Με τη μέθοδο **checkButtons()** του αντικειμένου **myButtons** καταχωρούμε σε μία μεταβλητή ακεραίου το μοναδικό αριθμό του αντικειμένου που πατήθηκε. Έτσι με ένα απλό έλεγχο προβαίνουμε σε ενέργεια. Εν προκειμένω, αφού το μοναδικό κουμπί που υπάρχει είναι το κουμπί **"menu",** μας παραπέμπει στην επόμενη συνάρτηση-οθόνη το **MainMenu().**



Σχήμα 6-1 Main Screen View

Στο τέλος της **while(1),** στις γραμμές 39 και 40, υπάρχουν δύο συναρτήσεις, η **readDHT()** και η **ReadSoil().**
Η πρώτη διαβάζει τη θερμοκρασία και την υγρασία περιβάλλοντος και η δεύτερη, διαβάζει την υγρασία χώματος των δυο χώρων. Ειδικότερα παρατίθενται οι κώδικες των συναρτήσεων.

```
1) void readDHT(){
2) Humid = dht.readHumidity();
3) Temp = dht.readTemperature();
4) if (isnan(Temp) || isnan(Humid)) {
```

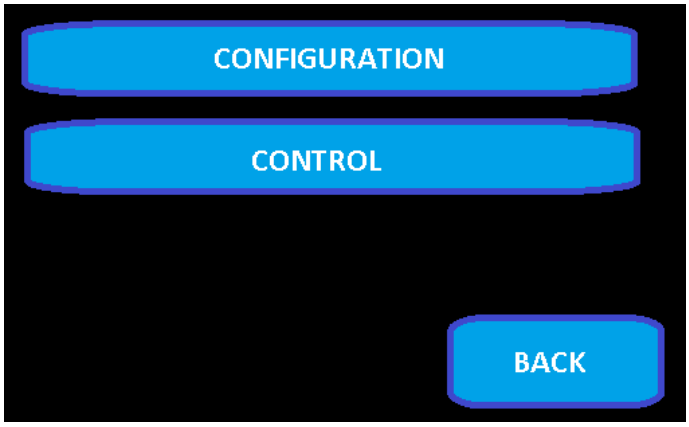
```
5) Serial.println(F("Failed to read from DHT"));
6) }
7) else {
8) delay(50);
9) Serial.print(F("Humidity: "));
10) Serial.print(Humid);
11) Serial.print(F(" %\t"));
12) Serial.print(F("Temperature: "));
13) Serial.print(Temp);
14) Serial.println(F(" *C"));
15) }
16) }
```

Αρχικά, Έχουμε δηλώσει το αντικείμενο **dht** της κλάσης **DHT** με ορίσματα την ψηφιακή θύρα που έχουμε συνδέσει το τσιπ, και τον τύπο του τσιπ(**DHT dht(DHTPIN, DHTTYPE);**)
Στις γραμμές 2 και 3 της συνάρτησης, διαβάζουμε και καταχωρούμε τις τιμές της υγρασίας και θερμοκρασίας, με τη χρήση των μεθόδων **readHumidity()** και **reatTemperature().** Αφού ελέγξουμε ότι η μέτρηση υπάρχει τότε την τυπώνουμε στη σειριακή θύρα.

```
1) void ReadSoil(){
   a) // read the input on analog pin 0:
2) int sensorValue;
3) int sensorValue2;
4) sensorValue = analogRead(A15);
5) sensorValue2=analogRead(A14);
6) sensorValue = constrain(sensorValue, 485, 1023);
7) sensorValue2 = constrain(sensorValue2, 485, 1023);
8) // print out the value you read:
9) //Serial.println(sensorValue);
```

```
10) //map the value to a percentage
11) Soil = map(sensorValue, 485, 1023, 100, 0);
12) Soil2 = map(sensorValue2, 485, 1023, 100, 0);
13) // print out the soil water percentage you calculated:
14) Serial.print(Soil);
15) Serial.println(F("%"));
16) Serial.print(Soil2);
17) Serial.println(F("%"));
18) delay(500);       // delay in between reads for stability
19) }
```

Εδώ, με δύο απλά **analogRead(AnalogPort)** διαβάζουμε τις τιμές των αισθητήρων υγρασίας και τις καταχωρούμε. Με τη συνάρτηση **constrain(x,a,b)** περιορίζουμε το αποτέλεσμα για την αποφυγή σφαλμάτων μέτρησης και με τη συνάρτηση **map()** μετατρέπουμε τη μέτρηση αυτή σε ποσοστό. Τέλος τυπώνουμε τα αποτελέσματα στη σειριακή θύρα.
Προχωρώντας στην επόμενη οθόνη, την **MainMenu(),** έχουμε δύο επιλογές. Να αλλάξουμε τις ρυθμίσεις του συστήματος ή να προβούμε σε έλεγχο του ποτίσματος. Αναλυτικότερα.



Σχήμα 6-2 MainManu View

Το κουμπί **CONFIGURATION** μας παραπέμπει στην επόμενη οθόνη στην οποία θα έχουμε περεταίρω επιλογές παραμετροποίησης. Το κουμπί **CONTROL,** μας παραπέμπει στην οθόνη ελέγχου του ποτίσματος. Το κουμπί **BACK,** μας παραπέμπει στην αμέσως προηγούμενη οθόνη. Όπως και η πρώτη συνάρτηση γραφικής απεικόνισης (**MainScreen()**) έτσι και οι επόμενες λειτουργούν με τον ίδιο ακριβώς τρόπο. Παρατίθεται ο κώδικας της **MainMenu().**

```
1)  void MainMenu(){
2)  myGLCD.clrScr();
3)  int Params,pressed_button1,Control,Back=NULL;//Info
4)  Params=myButtons.addButton(10,20,300,30,"CONFIGURATION");
5)  Control=myButtons.addButton(10,60,300,30,"CONTROL");
6)  Back=myButtons.addButton(200,199,100,40,"BACK");
7)  myButtons.drawButton(Params);
8)  myButtons.drawButton(Back);
9)  myButtons.drawButton(Control);
10) while(1){
11) if(myTouch.dataAvailable()){
    a) pressed_button1=myButtons.checkButtons();
    b) if(pressed_button1==Back){
    c) myButtons.deleteAllButtons();
    d) MainScreen();
    e) }
    f) else if(pressed_button1==Params){
    g) myButtons.deleteAllButtons();
    h) Parameters();
    i) }
    j) else if(pressed_button1==Control){
    k) myButtons.deleteAllButtons();
    l) ValveControl();
    m) }
12) }
13) }
14) }
```

Με το κουμπί **CONTROL** ερχόμαστε στην οθόνη **ValveControl().** Είναι η οθόνη στην οποία μπορούμε να ξεκινήσουμε ή να σταματίσουμε το ποτισμα του πρώτου ή του δεύτερου χώρου.

Σχήμα 6-3 ValveControl View

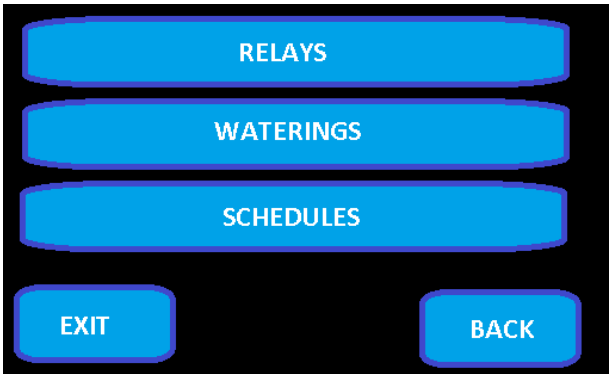
Με το κουμπί **TOGLE AREA 1** και **TOGLE AREA 2** ανοίγουμε ή κλείνουμε το ρελέ του κάθε χώρου. Με το κουμπί **EXIT** επιστρέφουμε στην πρώτη οθόνη. Το κουμπί **BACK** λειτουργεί ομοίως με το παραπάνω. Παρατίθεται ο κώδικας της συνάρτησης.

```
1)  void ValveControl(){
2)  myGLCD.clrScr();  int Koumpi,Exit,Back,Area1,Area2;
3)  Area1=myButtons.addButton(10,20,300,30,"Togle Area 1");
4)  Area2=myButtons.addButton(10,60,300,30,"Togle Area 2");
5)  Exit=myButtons.addButton(10,199,100,40,"EXIT");
6)  Back=myButtons.addButton(200,199,100,40,"BACK");
7)  myButtons.drawButton(Back);
8)  myButtons.drawButton(Exit);
9)  myButtons.drawButton(Area1);
10) myButtons.drawButton(Area2);
11) while(1){
12) if(myTouch.dataAvailable()){
    a) Koumpi=myButtons.checkButtons();
    b) if(Koumpi==Area1){
    c) if(Status=="Live"){
        i)  relay.off(1);
        ii) Status="Idle";
    d) }
    e) else if(Status=="Idle"){
        i)  relay.on(1);
        ii) Status="Live";
    f) }
    g) }
    h) else if(Koumpi==Area2){
    i) if(Status2=="Live"){
        i)  relay.off(2);
        ii) Status2="Idle";
    j) }
    k) else if(Status2=="Idle"){
        i)  relay.on(2);
        ii) Status2="Live";
    l) }
    m) }
    n) else if(Koumpi==Back){
    o) myButtons.deleteAllButtons();
    p) MainMenu();
    q) }
    r) else if(Koumpi==Exit){
    s) myButtons.deleteAllButtons();
    t) MainScreen();
```

```
    u)  }
13) }
14) }
15) }
```

Εδώ η συνάρτηση, σε περίπτωση συμβάντος, ελέγχει αν το εκάστοτε ρελέ είναι ανοικτό και αν ναι το κλείνει και αντίστροφα. Αυτό επιτυγχάνεται με τη χρήση δύο μεταβλητών, της **Status** και της **Status2.**

Στην επόμενη οθόνη(**Parameters())** στην οποία μας παραπέμπει το κουμπί **CONFIGURATION** έχουμε μία λίστα επιλογών, οθονών στην πραγματικότητα, για την παραμετροποίηση του συστήματος.



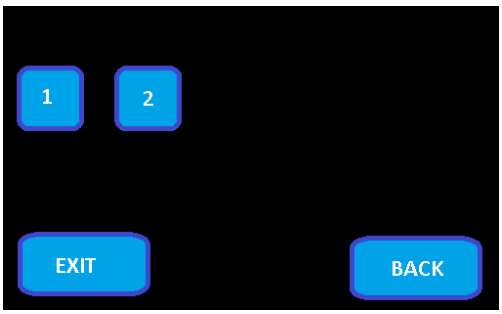
Σχήμα 6-4 Params View

Ομοίως με παραπάνω, κάθε κουμπί μας παραπέμπει σε μία διαφορετική οθόνη(συνάρτηση γραφικής απεικόνισης). Ειδικότερα παρατίθεται ο κώδικας της συνάρτησης.

```
1)  void Parameters(){
2)  myGLCD.clrScr();
3)  myGLCD.clrScr();
4)  int Relays,Waterings,schedules,pressed_button2,Back,Exit=NULL;
5)  Relays=myButtons.addButton(10,20,300,30,"RELAYS");
6)  Waterings=myButtons.addButton(10,60,300,30,"WATERINGS");
7)  schedules=myButtons.addButton(10,100,300,30,"SCHEDULE");
8)  Exit=myButtons.addButton(10,199,100,40,"EXIT");
9)  Back=myButtons.addButton(200,199,100,40,"BACK");
10) myButtons.drawButtons();
11) while(1){
12) if(myTouch.dataAvailable()){
    a)  pressed_button2=myButtons.checkButtons();
    b)  if(pressed_button2==Relays){
    c)  myButtons.deleteAllButtons();
    d)  Relay();
    e)  }
    f)  else if(pressed_button2==Waterings){
    g)  myButtons.deleteAllButtons();
    h)  Waters();
    i)  }
    j)  else if(pressed_button2==Back){
    k)  myButtons.deleteAllButtons();
    l)  MainMenu();
    m)  }
    n)  else if(pressed_button2==Exit){
```

```
    o) myButtons.deleteAllButtons();
    p) MainScreen();
    q) }
    r) else if(pressed_button2==schedules){
    s) myButtons.deleteAllButtons();
    t) edit_schedule();
    u) }
13) }
14) }
15) }
```

Το κουμπί **RELAYS** μας παραπέμπει στη συνάρτηση **Relay()** η οποία έχει την παρακάτω εμφάνιση.



Σχήμα 6-5 Relays View

Εδώ, επιλέγουμε αν θα έχουμε ένα χώρο ή δύο. Δηλαδή αν θα χρησιμοποιήσουμε ένα ρελέ ή δύο. Πατώντας το κουμπί **1** το σύστημα αποθηκεύει σε ένα αρχείο ότι θα έχουμε ένα ρελέ. Ομοίως πράττει και το κουμπί **2.** Τα κουμπιά **EXIT** και **BACK** έχουν τις ίδιες ιδιότητες όπως παραπάνω. Ειδικότερα παρατίθεται ο κώδικας της συνάρτησης.

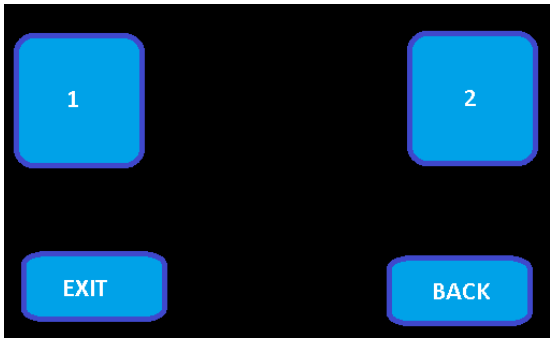
```
1)  void Relay(){
2)  myGLCD.clrScr();
3)  int Exit,Back,a,b,c,d,e,f,g,h,pressed_button3=NULL;
4)  Exit=myButtons.addButton(10,199,100,40,"EXIT");
5)  Back=myButtons.addButton(200,199,100,40,"BACK");
6)  a=myButtons.addButton(5,60,30,30,"1");
7)  b=myButtons.addButton(45,60,30,30,"2");
8)  myButtons.drawButtons();
9)  while(1){
10) if(myTouch.dataAvailable()){
    a) pressed_button3=myButtons.checkButtons();
    b) if(pressed_button3==a){
    c) myGLCD.print("You Have selected 1 Relay Output",10,110);
    d) writeFile("Relays.TXT","1");
    e) }
    f) else if(pressed_button3==b){
    g) myGLCD.print("You Have selected 2 Relay Output",10,110);
    h) writeFile("Relays.TXT","2");
    i) }
    j) else if(pressed_button3==Back){
    k) myButtons.deleteAllButtons();
    l) Parameters();
    m) }
```

```
    n) else if(pressed_button3==Exit){
    o) myButtons.deleteAllButtons();
    p) MainScreen();
    q) }
11) }
12) }
```

Εδώ η συνάρτηση αποκτά αρκετό ενδιαφέρον. Στη γραμμή 10d και 10h εκτελείται η συνάρτηση **writeFile(filename, data)** η οποία παίρνει τα δεδομένα που της δίνουμε ως όρισμα και τα γράφει σε ένα αρχείο (.TXT) κειμένου. Ειδικότερα παρατίθεται ο κώδικας της συνάρτησης.

```
1)  void writeFile(char * filename,char *data){
2)  if (file.exists(filename))
3)  file.delFile(filename);
4)  file.create(filename);
5)  res=file.openFile(filename, FILEMODE_TEXT_WRITE);
6)  if (res==NO_ERROR)
7)  {
8)  file.writeLn(data);
9)  delay(500);
10) file.closeFile();
11) }
12) else
13) {
14) switch(res)
15) {
    a) case ERROR_ANOTHER_FILE_OPEN:
    b) Serial.println(F("** ERROR: Another file is already open..."));
    c) break;
    d) default:
    e) Serial.print(F("** ERROR: "));
    f) Serial.println(res, HEX);
    g) break;
16) }
17) }
18) Serial.println(F("***** All done... *****"));
19) }
```

Εδώ η συνάρτηση αρχικά ελέγχει αν υπάρχει το αρχείο με τη χρήση της μεθόδου **exists(filename)** (γραμμή 2), και αν ναι το σβήνει με τη χρήση της μεθόδου **delFile(filename)** (γραμμή 3)**.** Έπειτα, αφού ανοίξει το αρχείο με τη χρήση της μεθόδου **openFile(filename, TEXTMODE)** (γραμμή 5) και ελέγξει ότι δεν υπάρχουν σφάλματα(γραμμή 6), τότε θα γράψει τα δεδομένα στο αρχείο με τη χρήση της μεθόδου **writeLn(data)** (γραμμή 8) και θα κλείσει το αρχείο για την αποφυγή υπερχειλίσεων μνήμης (**memory overflow)** με τη χρήση της μεθόδου **closeFile()** (γραμμή 10)**.** Αν υπάρξουν σφάλματα στο άνοιγμα του αρχείου, τότε θα τα τυπώσει στη σειριακή θύρα.
Περεταίρω, στην οθόνη **Waters()** στην οποία μας παραπέμπει το κουμπί **WATERINGS** έχουμε ακριβώς την ίδια λειτουργία.

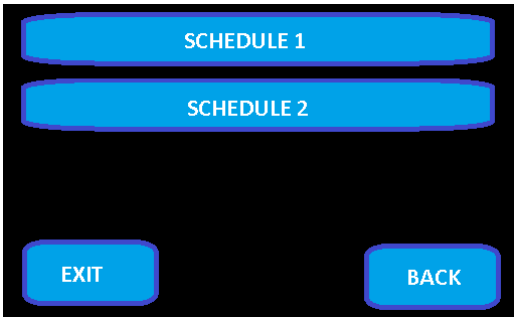


Σχήμα 6-6 Waterings View

Επιλέγοντας το κουμπί **1,** το σύστημα ρυθμίζεται για ένα πότισμα ημερησίως. Αντίστοιχα με το κουμπί **2** το σύστημα ρυθμίζεται για δύο ποτίσματα. Ομοίως το σύστημα αποθηκεύει τη ρύθμιση αυτή σε ένα αρχείο με τον ίδιο τρόπο όπως παραπάνω. Ειδικότερα παρατίθεται ο κώδικας της συνάρτησης. Τα κουμπιά **BACK** και **EXIT** έχουν τις ίδιες λειτουργίες με παραπάνω.

```
1)  void Waters(){
2)  myGLCD.clrScr();
3)  int Exit,Back,save,a,b,c,pressed_button4=NULL;
4)  Exit=myButtons.addButton(10,199,100,40,"EXIT");
5)  Back=myButtons.addButton(210,199,100,40,"BACK");
6)  save=myButtons.addButton(115,199,90,40,"SAVE");
7)  a=myButtons.addButton(10,30,80,80,"1");
8)  b=myButtons.addButton(235,30,80,80,"2");
9)  myButtons.drawButtons();
10) while(1){
11) if(myTouch.dataAvailable()){
    a)  pressed_button4=myButtons.checkButtons();
    b)  if(pressed_button4==a){
    c)  myGLCD.print("You Have selected 1 Watering",10,140);
    d)  *wateringsString='1';
    e)  }
    f)  else if(pressed_button4==b){
    g)  myGLCD.print("You Have selected 2 Waterings",10,140);
    h)  *wateringsString='2';
    i)  }
    j)  else if(pressed_button4==Back){
    k)  myButtons.deleteAllButtons();
    l)  Parameters();
    m)  }
    n)  else if(pressed_button4==Exit){
    o)  myButtons.deleteAllButtons();
    p)  MainScreen();
    q)  }
    r)  else if(pressed_button4==save){
    s)  writeFile(waterings,wateringsString);
    t)  }
12) }
13) }
14) }
```

Το κουμπί **SCHEDULES** της οθόνης **Parameters()** μας παραπέμπει σε μία λίστα επιλογών(οθονών) την οθόνη **edit_schedule()**.

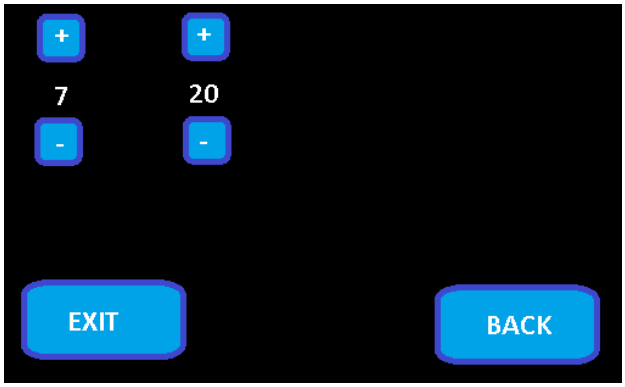


Σχήμα 6-7 Edit_schedule View

Ειδικότερα παρατίθεται ο κώδικας της συνάρτησης.

```
1)  void edit_schedule(){
2)  myGLCD.clrScr();
3)  int Exit,Back,schedule_1,schedule_2,pressed_button5=NULL;
4)  schedule_1=myButtons.addButton(10,20,300,30,"SCHEDULE 1");
5)  schedule_2=myButtons.addButton(10,60,300,30,"SCHEDULE 2");
6)  Exit=myButtons.addButton(10,199,100,40,"EXIT");
7)  Back=myButtons.addButton(200,199,100,40,"BACK");
8)  myButtons.drawButtons();
9)  while(1){
10) if(myTouch.dataAvailable()){
    a)  pressed_button5=myButtons.checkButtons();
    b)  if(pressed_button5==Back){
    c)  myButtons.deleteAllButtons();
    d)  Parameters();
    e)  }
    f)  else if(pressed_button5==Exit){
    g)  myButtons.deleteAllButtons();
    h)  MainScreen();
    i)  }
    j)  else if(pressed_button5==schedule_1){
    k)  myButtons.deleteAllButtons();
    l)  schedule1_screen();
    m)  }
    n)  else if(pressed_button5==schedule_2){
    o)  myButtons.deleteAllButtons();
    p)  schedule2_screen();
    q)  }
11) }
12) }
13) }
```

Τα κουμπιά **SCHEDULE 1** και **SCHEDULE 2** μας παραπέμπουν σε δύο οθόνες στις οποίες ρυθμίζουμε την ώρα του εκάστοτε ποτίσματος.



Σχήμα 6-8 Schedules View

Με τα κουμπιά "+" και "-" αυξομειώνουμε την ώρα και τα λεπτά. Ομοίως με παραπάνω η συνάρτηση αποθηκεύει σε ένα αρχείο τις ρυθμίσεις. Τα κουμπιά **BACK** και **EXIT** έχουν τις ίδιες λειτουργίες με παραπάνω. Και οι δύο οθόνες είναι ίδιες και έχουν ακριβώς την ίδια λειτουργία. Ειδικότερα παρατίθεται ο κώδικας της πρώτης συνάρτησης(**schedule1_screen()**).

```
1)  void schedule1_screen(){
2)  myGLCD.clrScr();
3)  int Exit,Back,a,b,c,d,e,f,g,h,save,plus1,plus2,minus1,minus2,pressed_button7=NULL;
4)  Exit=myButtons.addButton(10,199,100,40,"EXIT");
5)  Back=myButtons.addButton(210,199,100,40,"BACK");
6)  save=myButtons.addButton(115,199,90,40,"SAVE");
7)  plus1=myButtons.addButton(10,10,30,30,"+");
8)  plus2=myButtons.addButton(60,10,30,30,"+");
9)  minus1=myButtons.addButton(10,60,30,30,"-");
10) minus2=myButtons.addButton(60,60,30,30,"-");
11) myButtons.drawButtons();
12) while(1){
    a) myGLCD.printNumI(int(h1),15,45);
    b) myGLCD.printNumI(int(m1),65,45);
    c) if(myTouch.dataAvailable()){
    d) pressed_button7=myButtons.checkButtons();
    e) if(pressed_button7==Back){
    f) myButtons.deleteAllButtons();
    g) edit_schedule();
    h) }
    i) else if(pressed_button7==Exit){
    j) myButtons.deleteAllButtons();
    k) MainScreen();
    l) }
    m) else if(pressed_button7==plus1){
    n) if(h1<24)
        i) h1++;
    o) }
    p) else if(pressed_button7==plus2){
    q) if(m1<60)
        i) m1++;
    r) }
    s) else if(pressed_button7==minus1){
```

```
t)  if(h1>0)
    i)  h1--;
u)  }
v)  else if(pressed_button7==minus2){
w)  if(m1>0)
    i)  m1--;
x)  }
y)  else if(pressed_button7==save){
z)  String str=String(h1)+','+String(m1);
aa) str.toCharArray(scheduleString1,20);
bb) writeFile(filenameS1,scheduleString1);
cc) }
dd) }
13) }
14) }
15)
```

6.4 Δημοσίευση Δεδομένων

Όπως προαναφέραμε, το σύστημα περιοδικά δημοσιεύει τις μετρήσεις και την κατάστασή του, σε έναν απομακρυσμένο διακομιστή(sever). Ο τρόπος με τον οποίο το σύστημα αποστέλλει τα δεδομένα του στο server, είναι με τη μέθοδο του Http Post. Δηλαδή με την αποστολή ενός Http αιτήματος, ακολουθούμενο από τις ανάλογες μεταβλητές και τις τιμές τους. Από τη μεριά του ο server , θα πρέπει να διαθέτει ένα web server(Apache) για να παραλάβει το αίτημα και να το αξιοποιήσει και επιπρόσθετα μία βάση δεδομένων(MySQL Server) για να αποθηκεύσει τα δεδομένα.
Για την υλοποίηση αυτή, χρησιμοποιήσαμε μία πλατφόρμα δωρεάν και ανοικτού κώδικα, το WAMP. Το WAMP είναι ακρωνύμιο των λέξεων Windows Apache MySQL PHP και ενσωματώνει αυτά τα τρία πιο απαραίτητα χαρακτηριστικά που πρέπει να διαθέτει ένας server. Τον Apache server που θα είναι ο dedicated web Server, το MySQL server που θα χειρίζεται τη βάση δεδομένων και φυσικά την PHP τη script γλώσσα που θα αναλαμβάνει την αλληλεπίδραση των δύο παραπάνω.

Σχήμα 6-9 WAMP Server

6.4.1 BackEnd

Βάση Δεδομένων

Για τη διατήρηση της απλότητας του συστήματος, η βάση δεδομένων μας θα είναι εξίσου απλή. Δηλαδή, η βάση δεδομένων θα έχει όνομα "**arduino**" και θα περιέχει μόνο ένα table με όνομα "**sensors**". Και θα περιέχει τις στήλες

Column Name	Type	Description	Collation	Primary Key
Temp	Text	Ambient Temperature	utf8_general	No
Humid	Text	Ambient Humidity	utf8_general	No
Soil	Text	Soil Humidity for Area1	utf8_general	No
Soil2	Text	Soil Humidity for Area 2	utf8_general	No
Status	Text	Relay Status for Area 1	utf8_general	No
Sattaus2	Text	Relay Status for Area 2	utf8_general	No
Date	DateTime	Timestamp		No
indicator	AutoIncrement	Id	utf8_general	Yes

Πίνακας 6-1 sensors

Web Server

Στην ουσία ο server, μετά την εγκατάσταση της πλατφόρμας, είναι έτοιμος. Το μόνο που χρειάζεται είναι μία σταθερή διεύθυνση IP για να μπορεί κάποιος να τον προσπελάσει απομακρυσμένα. Στον φάκελο εγκατάστασης του server (c:\wamp) περιέχεται ο φάκελος ο οποίος θα είναι δημόσιος στο δίκτυο, ο "c:\wamp\www". Εκεί μέσα τοποθετούνται όλα τα script και οι ιστοσελίδες που θέλουμε να προσπελαύνονται απομακρυσμένα. Ο server είναι αρχικά ρυθμισμένος να κάνει listen, δηλαδή να αναμένει τα αιτήματα, στην πόρτα 80 και οι χρήστες(guests) έχουν δικαιώματα μόνο για ανάγνωση. Εάν κάποιος θελήσει να αλλάξει τις αρχικές ρυθμίσεις, θα πρέπει να παραμετροποιήσει το αρχείο ρυθμίσεων του server, **httpd.conf** το οποίο βρίσκεται στο φάκελο "C:\wamp\bin\apache\Apache2.4.4\conf\" **.** Εκεί περιέχονται όλες οι παράμετροι ασφάλειας του server.
Περεταίρω, για να μπορέσει κάποιος να συνδεθεί στο server και να πάρει ή να εισάγει δεδομένα στον MySQL server, δεν μπορεί να γίνει απευθείας. Θα πρέπει πρώτα να συνδεθεί με τον Apache, και με κάποιο τρόπο, να κάνει το ανάλογο «ερώτημα» από τον MySQL server. Τη διαμεσολάβηση αυτή, αναλαμβάνει η PHP, με την τοποθέτηση συγκεκριμένων script αρχείων στο δημόσιο φάκελο.

PHP

Η php, είναι μία script γλώσσα προγραμματισμού για την κατασκευή δυναμικών ιστοσελίδων, όμως δεν είναι γλώσσα "markup". Για την τελική απεικόνιση του αποτελέσματος χρειάζεται η γλώσσα HTML(**hypertext markup Language**). Παρόλα αυτά, μπορεί αν υπάρξει και αυτόνομα, για την πραγματοποίηση κάποιον ενεργειών οι οποίες δεν χρειάζεται να έχουν οπτικό αποτέλεσμα όπως πχ την εισαγωγή δεδομένων σε μία βαση(**"insert into database…"**). Για τις δικές μας ανάγκες, θα χρησιμοποιήσουμε δύο script. Το ένα θα είναι για να απαντάει στα ερωτήματα του συστήματος και το άλλο θα απαντάει στα ερωτήματα της android εφαρμογής. Ειδικότερα παρατίθενται οι κώδικες των script.

Sensors.php

```
1)  <?php
2)  //Connect to database
3)  $date = new DateTime();
4)  $date->setTimezone(new DateTimeZone('Europe/Athens'));
5)  $fdate = $date->format('Y-m-d H:i:s'); // same format as NOW()
6)  $con = mysql_connect("localhost", "username", "password");
7)  if(!$con)
    a) die('Could not connect: ' .mysql_error());
8)  mysql_select_db("arduino", $con);
9)  $result = mysql_query("INSERT INTO sensors(Temp,Humid,Soil,Soil2,Status,Status2,Date) VALUES
    ('".$_GET['Temp']."','".$_GET['Humid']."','".$_GET['Soil']."','".$_GET['Soil2']."','".$_GET['Status']."','".$_GE
    T['Status2']."','$fdate')");
10) mysql_close($con);
11) ?>
```

Αυτό είναι το script που εξυπηρετεί το σύστημα. Εδώ αρχικά ρυθμίζουμε την ώρα στη ζώνη ώρας και την τοποθετούμε στη μεταβλητή $fdate(γραμμή 4-5).
Έπειτα, επιχειρούμε να συνδεθούμε στη βάση δεδομένων(γραμμή 6), αν υπαρχει επιτυχία τότε εισάγουμε στη βάση τις τιμές(γραμμή 9). Όπως προαναφέραμε, το σύστημα κάνει ένα http αίτημα(post) της μορφής **http://serverAddress:port/arduino/sensors.php?Temp=34&Humid=86&Soil=43&Soil2=23&Status=Idle&Status2=Live**
Έτσι λοιπόν η PHP χρησιμοποιεί τη μέθοδο **"$_GET["variableName"]** για να αρπάξει τις τιμές των μεταβλητών. Τέλος κλείνει τη σύνδεση(γραμμή 10).

Android.php

```
1)  <?php
2)  mysql_connect("localhost","username","password");
3)  mysql_select_db("arduino");
4)  $sql=mysql_query("SELECT Temp, Humid, Soil, Soil2, Status, Status2, Date FROM sensors ORDER BY Date
    DESC limit 0,1");
5)  while($row=mysql_fetch_assoc($sql))
6)  $output[]=$row;
7)  print(json_encode($output));// this will print the output in json
8)  mysql_close();
9)  ?>
```

Αυτό είναι το script που εξυπηρετητή την android εφαρμογή. Εδώ πάλι συνδέεται στη βάση, κάνει ένα ερώτημα **select(**γραμμή 4**),** και το αποτέλεσμα του ερωτήματος, το τυπώνει σε κωδικοποίηση **json.** Η **json** (**javaScript Object Notation)** κωδικοποίηση είναι ένα πρότυπο ανοικτού κώδικα το οποίο χρησιμοποιείται για την εύκολη ανταλλαγή δεδομένων μεταξύ server και εφαρμογών. Βασίζεται στη γλώσσα Javascript όμως δρα αυτόνομα. Είναι ένα πρότυπο το οποίο είναι εύκολο για το χρήστη να το γράψει και εύκολο για το μηχάνημα να το μεταγάγει και να το παράγει. Το αποτέλεσμα που παράγει είναι της μορφής

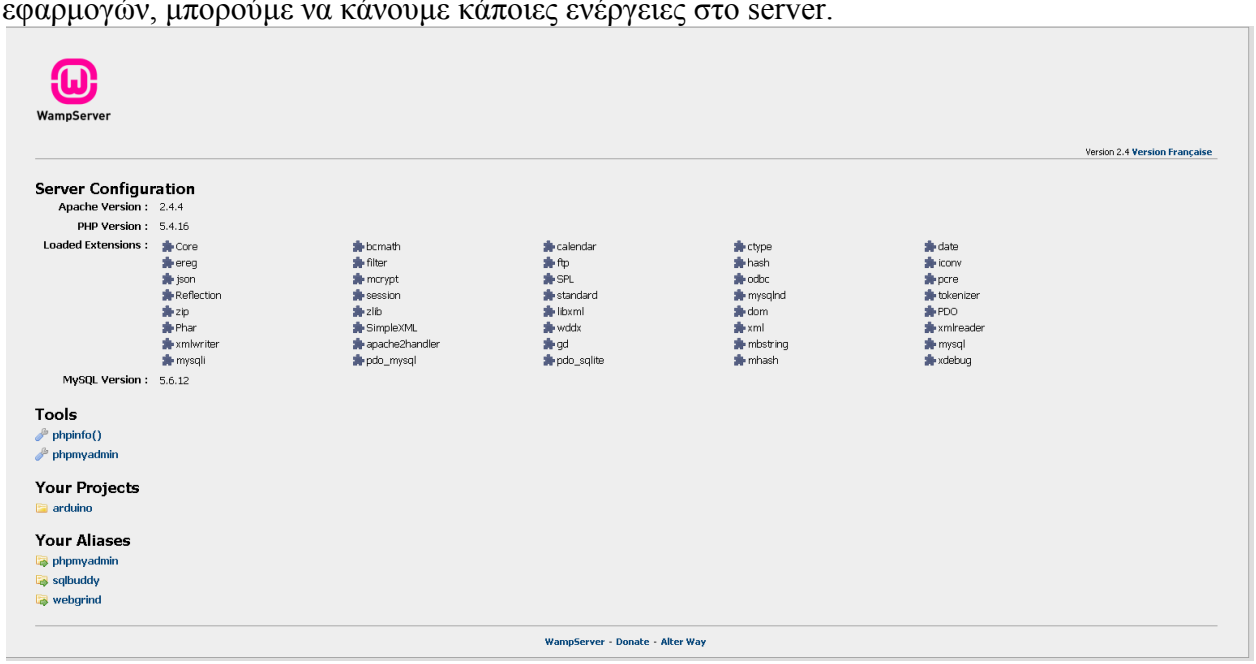
```
[{"Temp":"34","Humid":"86","Soil":"43","Soil2":"23","Status":"Idle","Status2":"Live","Date":"2014-08-21
17:49:24"}]
```

Φυσικά το πρότυπο αυτό υλοποιείται πλήρως και στο Android SDK.

Σχήμα 6-10 JSON

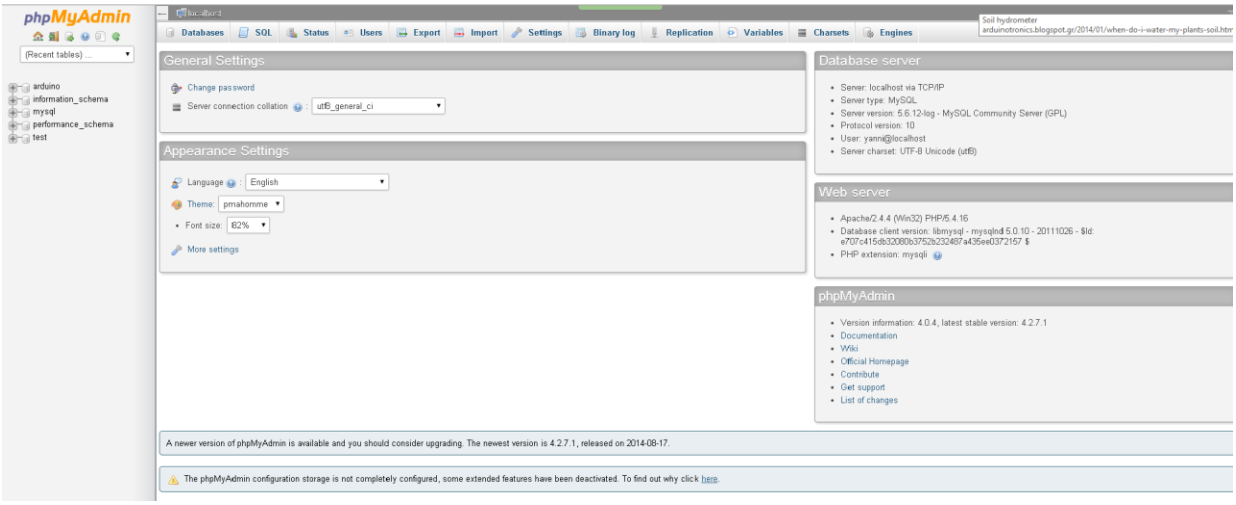
6.4.2 FrontEnd

Η πλατφόρμα WAMP, μας προσφέρει ένα Panel μέσω του οποίου, με τη βοήθεια κάποιων εφαρμογών, μπορούμε να κάνουμε κάποιες ενέργειες στο server.



Σχήμα 6-11 Apache FrontEnd

Δυστυχώς οι μόνες διαθέσιμες εφαρμογές, αφορούν τον MySQL server. Αντίθετα με άλλα επαγγελματικά Panel, όπως πχ το **CPANEL**, τα οποία προσφέρουν πολλές παραπάνω δυνατότητες . Η εφαρμογή η οποία χρησιμοποιήσαμε για τη δημιουργία και παραμετροποίηση της βάσης δεδομένων, είναι η **“PhpMyAdmin”.** Είναι μία ελαφριά και εύκολη προς το χρήστη εφαρμογή, που καθιστά τη δημιουργία, το χειρισμό και την προσπέλαση μίας βάσης πολύ απλό



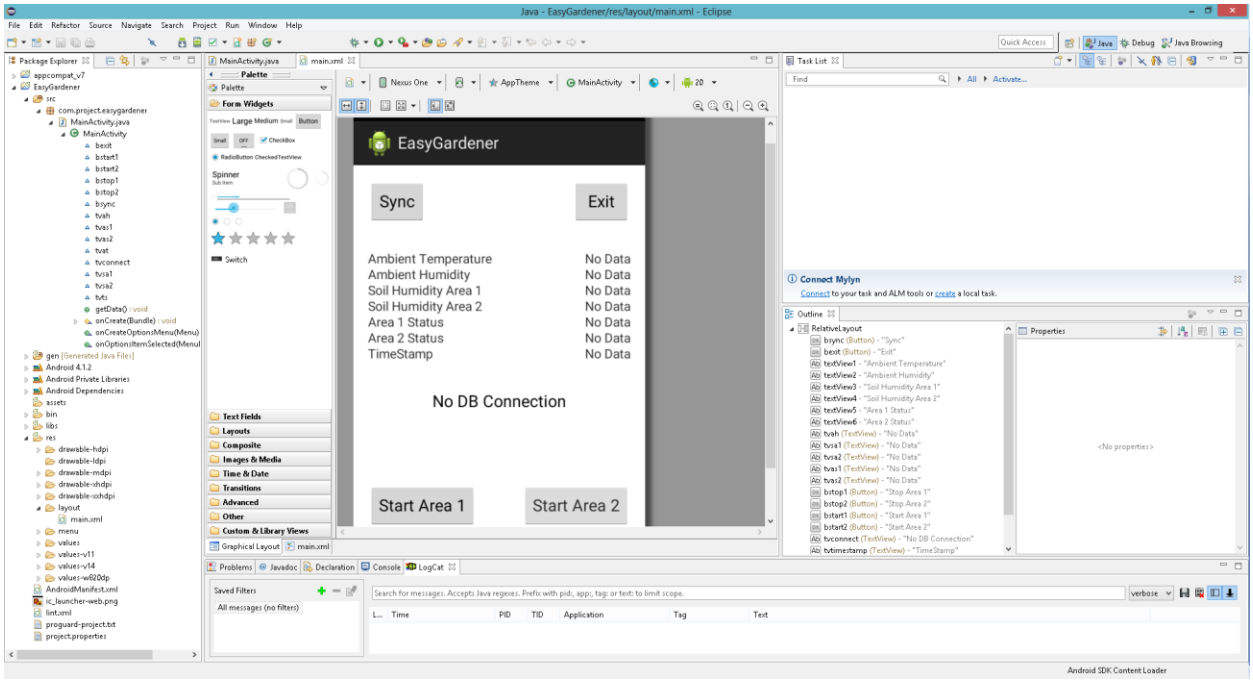
Σχήμα 6-12 PhpMyAdmin

6.5 Android

Το **Android** είναι ένα λειτουργικό σύστημα για συσκευές κινητής τηλεφωνίας το οποίο τρέχει τον πυρήνα του λειτουργικού Linux. Αρχικά αναπτύχθηκε από την Google και αργότερα από την Open Handset Alliance. Επιτρέπει στους κατασκευαστές λογισμικού να συνθέτουν κώδικα με την χρήση της γλώσσας προγραμματισμού Java, ελέγχοντας την συσκευή μέσω βιβλιοθηκών λογισμικού ανεπτυγμένων από την Google. Το προγραμματιστικό περιβάλλον και οι βιβλιοθήκες ανάπτυξης, προσφέρονται δωρεάν στους προγραμματιστές. Κάτι το οποίο καθιστά το ίδιο το λειτουργικό, όπως και την ανάπτυξη εφαρμογών σε αυτό πολύ προσφιλές.
Το σύστημα μας θα ελέγχεται απομακρυσμένα, από μία **android** εφαρμογή, η οποία θα είναι απλή και κατανοητή από τον οποιονδήποτε.

6.5.1 Προγραμματιστικά Εργαλεία

Το πιο διαδεδομένο δωρεάν εργαλείο ανάπτυξης κώδικα, ιδιαίτερα όσον αφορά τη JAVA και εν προκειμένω το Android, είναι η σουίτα λογισμικού **Eclipse.** Είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης, το οποίο με τα εργαλεία που διαθέτει, αυτόματη παραγωγή κώδικα, αποσφαλματωτή κτλ, καθιστά τον προγραμματισμό και την ανάπτυξη λογισμικού πολύ απλούστερη από ότι ήταν στο παρελθόν.



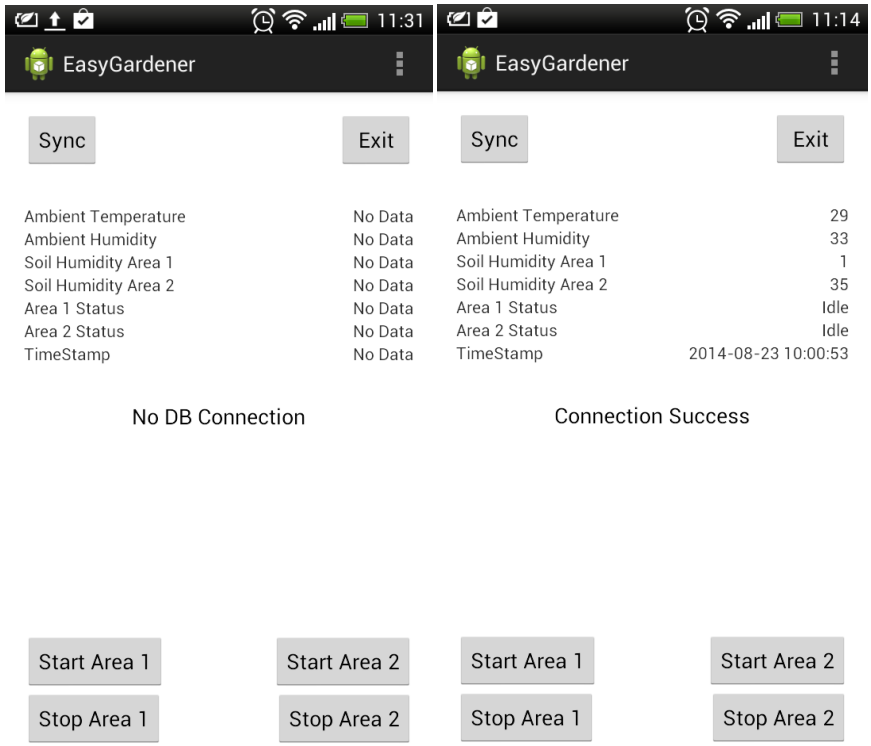
Σχήμα 6-13 Eclipse Workspace

6.5.2 *Προγραμματισμός*

Ο προγραμματισμός μίας εφαρμογής Android, γίνεται με τη χρήση του compiler και των βιβλιοθηκών της JAVA. Επομένως το περιβάλλον και η γενικότερη λογική, παραμένουν ακριβώς οι ίδιες. Ο προγραμματισμός σε ANDROID SDK, χωρίζεται σε τρία στάδια. Το **layout** της εφαρμογής, δηλαδή η εμφάνιση και τα αντικείμενα που πλαισιώνουν μία εφαρμογή, το **MainActivity,** το κομμάτι που περιέχει το σημαντικότερο κομμάτι του κώδικα και στο οποίο γίνεται η ολική υλοποίηση της εφαρμογής και τέλος το **AndroidManifest.xml** στο οποίο εντέλει ορίζονται τα δικαιώματα που θα έχει αυτή η εφαρμογή στο λειτουργικό σύστημα που θα τρέχει. Δηλαδή ποιες από τις δυνατότητες του Android(SMS, internet κτλ) θα μπορεί να χρησιμοποιεί.

6.5.3 *Η Εφαρμογή*

Η διεπαφή της εφαρμογής είναι σχεδιασμένη με γνώμονα την χρηστικότητα και την εύκολη κατανόηση των λειτουργιών της. Αποτελείται από μία οθόνη η οποία παρουσιάζει τις περιβαλλοντικές μετρήσεις και την κατάσταση του συστήματος, δεδομένα τα οποία ανασύρει από την απομακρυσμένη βάση δεδομένων και επιπλέον διαθέτει και κάποια κουμπιά χειρισμού.



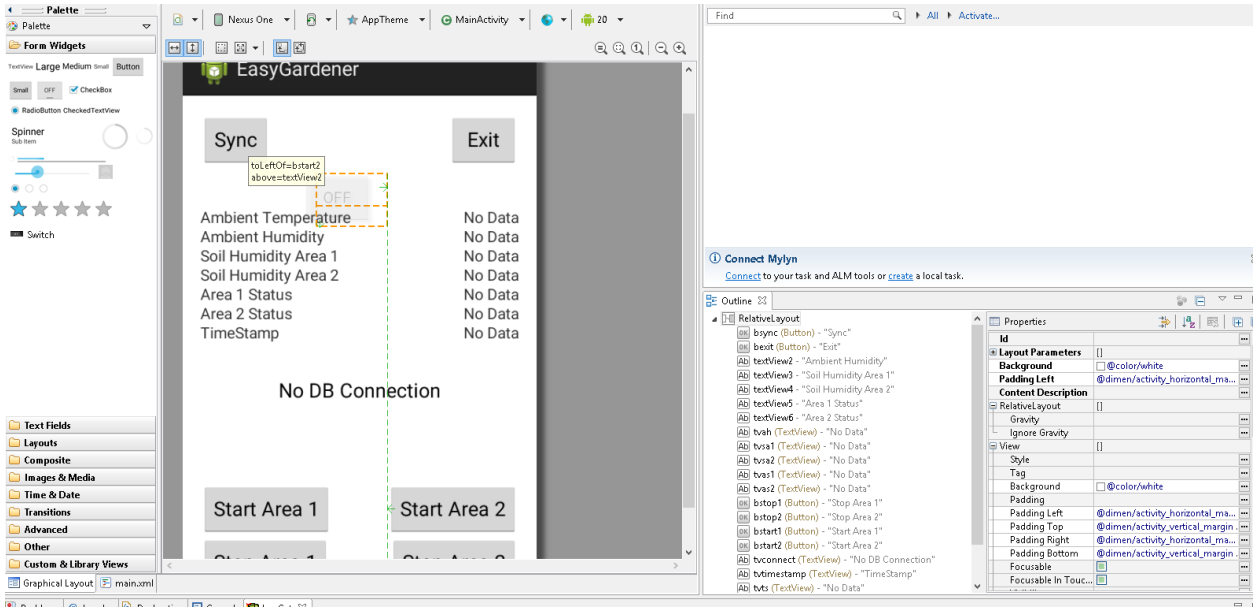
Σχήμα 6-14 Easy Gardener

Ειδικότερα, με το κουμπί "**Sync",** η εφαρμογή κάνει ένα ερώτημα **select** στη βάση δεομένων και αφού τα ταξινομήσει ανά ημερομηνία δημοσίευσης, διαλέγει την νεότερη και την εμφανίζει μπροστά στην οθόνη. Όταν η σύνδεση είναι επιτυχής Το TextView με κείμενο "No DB Connection" γίνεται "Connection Success". Με το κουμπί "**Start Area 1"** ξεκινάμε το πότισμα στον πρώτο χώρο, στέλνοντας ένα γραπτό μήνυμα της μορφής **"R1ON"** στο σύστημα. Αντίθετα με το κουμπί **"Stop Area 1"** σταματούμε το πότισμα στον πρώτο χώρο, στέλνοντας στο σύστημα ένα γραπτό μήνυμα της μορφής **"R1OFF".** Ομοίως λειτουργούν και τα άλλα δύο κουμπιά **«Start Area 2"** και **"Stop Area 2"**με τη διαφορά ότι το κείμενο των μηνυμάτων είναι **"R2ON"** και "**R2OFF".** Τέλος με το κουμπί **"Exit"** η εφαρμογή τερματίζεται.

6.5.4 Ο Κώδικας

Το layout

Πρωτίστως σε μία εφαρμογή πρέπει να σχεδιάσουμε τη διεπαφή της. Την εμφάνιση και τη στρατηγική τοποθέτηση των αντικειμένων που θα την πλαισιώνουν. Το Eclipse παρέχει ένα ισχυρό εργαλείο για την εύκολη πραγματοποίηση του σχεδιασμού. Παράλληλα, με τη γεννήτρια κώδικα το καθιστά ακόμη πιο γρήγορο.



Σχήμα 6-15 Main Layout Design

Με τη γραφική παρουσίαση των αντικειμένων, ο προγραμματιστής καταλαβαίνει ακριβώς τις ιδιότητες του και με ένα απλό drag and drop μέσα στη φόρμα, η πλατφόρμα το τοποθετεί στην οθόνη και γεννά αυτόματα τον κώδικα μέσα στο **Main.xml** που περιέχει όλα τα αντικείμενα της φόρμας μαζί με τις ιδιότητες που τα χαρακτηρίζουν. Στο αρχείο αυτό, υλοποιείται και περιγράφεται το layout της φόρμας. Το layout είναι ένα σύνολο κανόνων και παραδοχών, που αφορούν τη στοίχιση και τη γραφική τοποθέτηση των αντικειμένων μέσα στη φόρμα. Υπάρχει ένας αριθμός από διαφορετικά layout, ανάλογα τις απαιτήσεις της εκάστοτε εφαρμογής. Στη προκειμένη περίπτωση χρησιμοποιήσαμε ένα δυναμικό ευέλικτο προσαρμοστικό και σχετικά καινούριο layout, το **"Relative Layout"** . Ειδικότερα παρατίθεται ο κώδικας του **Main.xml** στο Παράρτημα στο τέλος της εργασίας**.**

Παρατηρούμε ότι το σύνολο των αντικειμένων, περιβάλει το layout. Μέσα σε αυτό τοποθετούνται και όλα τα αντικείμενα τα οποία «υπακούν» στους κανόνες του περιβάλλοντός τους. Ακόμη, τα κουμπιά δηλώνονται "<Button />" , οι ετικέτες δηλώνονται ως "<TextView/>" και μέσα τους, εσωκλείονται οι ιδιότητες τους. Επίσης παρατηρούμε ότι κάθε αντικείμενο, σαν πρώτη ιδιότητα, έχει ένα "**id".** Αυτό το μοναδικό "**id",** το καταχωρούμε στο αντικείμενο, ούτως ώστε να μπορούμε να το αναζητήσουμε μονοσήμαντα μέσα στη μνήμη του προγράμματος. Τέλος, μέσα στις ιδιότητες του κάθε αντικείμενο, περιέχει χωροταξικές ιδιότητες σε σχέση με τα γειτονικά του αντικείμενα.

MainActivity.java

Μέσα σε αυτό το αρχείο, υπάρχει η κύρια κλάση του project η **MainActivity** και η μέθοδος η οποία το υλοποιεί **onCreate.** Επίσης μέσα σε αυτό το αρχείο εισάγονται και όλες οι βιβλιοθήκες που θα χρησιμοποιήσουμε. Αρχικά εδώ, θα πρέπει να αντιστοιχίσουμε τα αντικείμενα της φόρμας με τις ανάλογες μεταβλητές, ούτος ώστε να μπορούμε να τα χειριστούμε. Πρώτα, πρέπει να δηλώσουμε μία μεταβλητή της κλάσης του αντικειμένου πχ **Button** για το κουμπί. Αφού

δηλώσουμε τη μεταβλητή, πρέπει να την αντιστοιχίσουμε με το ανάλογο αντικείμενο. Αυτό θα το επιτύχουμε με τη χρήση της μεθόδου **findViewById(id).** Το **id** είναι η μοναδική ονομασία που δώσαμε στο αντικείμενο μας μέσα στο **Main.xml.** Έπειτα, αν το αντικείμενο αυτό παράγει κάποια ενέργεια, είναι πχ κουμπί, θα πρέπει να υλοποιήσουμε τον ActionListener για το αντικείμενο αυτό. Θα πρέπει δηλαδή να υλοποιήσουμε την μέθοδο που θα αξιοποιήσει το συμβάν που θα παράγει το αντικείμενο σε περίπτωση ενεργοποίησης του, και θα προβεί σε κάποια ενέργεια, συνήθως στην εκτέλεση κάποιας μεθόδου.
Επί παραδείγματι, η μέθοδος αυτή για ένα κουμπί, έχει την παρακάτω μορφή.

```
1) bexit.setOnClickListener(new View.OnClickListener() {
        1. @Override
        2. public void onClick(View v) {
            i. // TODO Auto-generated method stub
            ii. System.exit(1);
        3. }
(b) });
```

Στην προκειμένη περίπτωση, το **"bexit"** το οποίο είναι αντικείμενο της κλάσης Button υλοποιεί τη μέθοδο "setOnClickListener()" η οποία παίρνει ως όρισμα ένα αντικείμενο "OnCliclLIstener" της κλάσης "View" το οποίο υλοποιεί τη μέθοδο "onClick() μέσα στην οποία τοποθετούμε οποιαδήποτε ενέργεια θέλουμε να προβεί το συγκεκριμένο αντικείμενο.
Αυτό, πραγματοποιείται για όλα τα αντικείμενα που μπορούν να παράγουν κάποια ενέργεια. Αυτά μπορεί να είναι κουμπιά radio button, μπάρα κύλισης κτλ, κάθε αντικείμενο με το δικό του σετ ενεργειών και μεθόδων, ανάλογα την κλάση τους. Βεβαίως στην υφιστάμενη εφαρμογή διαθέτουμε μόνο κουμπιά και TextView. Οι ActionListeners των αντικειμένων, υλοποιούνται μέσα στη κυρίως μέθοδο (**onCreate),** όμως όλες οι μέθοδοι δηλώνονται και υλοποιούνται έξω από αυτή.

6.5.5 Ενέργειες

Exit

Το κουμπί **Exit** έχει την πιο απλή λειτουργία της εφαρμογής μας. Το κουμπί αυτό τερματίζει την εφαρμογή και απελευθερώνει τη μνήμη που κατέλαβε με τη χρήση της μεθόδου **"exit(1)"** της κλάσης **"System".**

Start Area 1

Αυτό το κουμπί, ξεκινά το πότισμα του πρώτου χώρου με γραπτό μήνυμα "R1ON" και εμφανίζει ένα μήνυμα bubble ότι η αποστολή ήταν επιτυχής. Ειδικότερα παρατίθεται ο κώδικας.

```
1) bstart1.setOnClickListener(new View.OnClickListener() {
        1. @Override
        2. public void onClick(View v) {
            i. // TODO Auto-generated method stub
            ii. SmsManager manager=SmsManager.getDefault();
            iii. manager.sendTextMessage("6978063409", null, "R1ON", null, null);
            iv. Toast.makeText(getApplicationContext(), "Send Ok", Toast.LENGTH_LONG).show();
        3. }
(b) });
```

Δηλώνουμε μία μεταβλητή **manager,** της κλάσης **SmsManager (2.ii)** και έπειτα στέλνουμε SMS το κείμενο που θέλουμε στον εκάστοτε αριθμό με τη χρήση της μεθόδου **sendTextMessage()** . Τέλος, δημιουργούμε ένα bubble μήνυμα με τη μέθοδο **makeText** της κλάσης **Toast** και το εμφανίζω με τη μέθοδο **show.**
Ομοίως πράττουμε και για τα κουμπιά **Stop Area 1, Start Area 2,** και **Stop Area 2.** Με μοναδική διαφορά ότι στο κείμενο του SMS τοποθετούμε **R1OFF, R2ON** και **R2OFF** αντίστοιχα.

Sync

Σε αυτό το κουμπί εκτελούμε μία μέθοδο, η οποία είναι και η σημαντικότερη της εφαρμογής μας. Την **getData()(1.2.ii).**

```
1) bsync.setOnClickListener(new View.OnClickListener() {
        1. @Override
        2. public void onClick(View v) {
            i. // TODO Auto-generated method stub
            ii. getData();
        3. }
(b) });
```

Αρχικά εκτελεί ένα http αίτημα στο server ,που είναι αποθηκευμένο το php script που αναλαμβάνει το ερώτημα στη βάση, παίρνει την απάντηση, την μετατρέπει σε string, και την παρουσιάζει στην οθόνη μας. Ειδικότερα παρατίθεται ο κώδικας της **getData().**

```
1) public void getData(){
    i) String result = "";
    ii) InputStream isr = null;
    iii) try{
        (1) HttpClient httpclient = new DefaultHttpClient();
        (2) HttpPost httppost = new HttpPost("http://gnikoloudakis.dyndns.org:8081/arduino/android.php"); //YOUR PHP SCRIPT
            ADDRESS
        (3) HttpResponse response = httpclient.execute(httppost);
        (4) HttpEntity entity = response.getEntity();
        (5) isr = entity.getContent();
        (6) tvconnect.setText("Connection Success");
b) }
c) catch(Exception e){
        (1) Log.e("log_tag", "Error in http connection "+e.toString());
        (2) tvconnect.setText("Couldnt connect to database");
d) }
e) //convert response to string
f) try{
        (1) BufferedReader reader = new BufferedReader(new InputStreamReader(isr,"iso-8859-1"),8);
        (2) StringBuilder sb = new StringBuilder();
        (3) String line = null;
        (4) while ((line = reader.readLine()) != null) {
            (i) sb.append(line + "\n");
        (5) }
        (6) isr.close();

        (7) result=sb.toString();
g) }
h) catch(Exception e){
        (1) Log.e("log_tag", "Error  converting result "+e.toString());
i) }

j) //parse json data
k) try {
        (1) JSONArray jArray = new JSONArray(result);

        (2) for(int i=0; i<jArray.length();i++){
```

```
                (i)    JSONObject json = jArray.getJSONObject(i);
            (ii)   tvat.setText(json.getString("Temp"));
            (iii)  tvah.setText(json.getString("Humid"));
                (iv)   tvsa1.setText(json.getString("Soil"));
                (v)    tvsa2.setText(json.getString("Soil2"));
                (vi)   tvas1.setText(json.getString("Status"));
                (vii)  tvas2.setText(json.getString("Status2"));
                (viii) tvts.setText(json.getString("Date"));


            (3)   }

l)    } catch (Exception e) {
        i)    // TODO: handle exception
            (1)   Log.e("log_tag", "Error Parsing Data "+e.toString());
m)    }

n)    }
```

Παρατηρούμε τη χρήση μίας συγκεκριμένης δομής, σε κάθε σετ ενεργειών που ενδέχεται να παράγουν κάποιο σφάλμα. Αυτή η δομή είναι η **try/catch.** Εκτελούμε ένα σετ ενεργειών μέσα στα πλαίσια της **try{….} ,** και με την **catch(Exception e){…..}** χειριζόμαστε το τυχόν παραγόμενο σφάλμα. Με αυτή τη διαδικασία διευκολύνουμε πολύ το debugging.
Η μέθοδος χωρίζεται σε τρία μέρη. Στο πρώτο μέρος, εκτελείται το http αίτημα. Αναλυτικότερα, Δηλώνουμε μια μεταβλητή httpclient της κλάσης HttpClient. Αυτή η οντότητα θα εκτελέσει το αίτημα(1.iii.1). Στη συνέχεια δηλώνουμε μία μεταβλητή httppost της κλάσησς HttpPost που θα είναι το ίδιο το αίτημα (http://gnikoloudakis.dyndns.org:8081/arduino/android.php") το οποίο στην πραγματικότητα είναι η κλήση του αρχείου android.php που βρίσκεται στο server(1.iii2).
Μετά δηλώνουμε την response της κλάσης HttpResponse και της καταχωρούε την απάντηση του αιτήματος με τη χρήση της μεθόδου execute(httppost) (1.iii.3). Αμέσως μετά εξαγάγουμε από τηυν απάντηση το σώμα της με τη χρήση της μεθόδου getEntity() και την καταχωρούμε στη μεταβλητή entity της κλάσης HttpEntity(1.iii.4). Τέλος εξαγάγουμε το περιεχόμενο του entity με τη χρήση της μεθόδου getContent() και το καταχωρούμε στη μεταβλητή isr του τύπου InputStream και αλλάζουμε το κείμενο της ετικέτας με όνομα tvconnect σε Connection Success με τη χρήση της μεθόδου setText("…"). Αν το αίτημα αποτύχει, τότε η catch παίρνει το μήνυμα σφάλματος και το καταγράφει . Επιπλέον ξανά αλλάζει το κείμενο της tvconnect σε Couldn't connect to database.
Το δεύτερο σκέλος της μεθόδου, μετατρέπει την απάντηση σε String. Μέσα στη μεταβλητή reader της κλάσης BufferReader εισάγουμε με τη μέθοδο InputStreamReader το Input Stream isr με κωδικοποίηση iso-8859-1 (f.1). Έπειτα, δηλώνουμε μία μεταβλητή sb της κλάσης StringBuilder και μία μεταβλητή line της κλάσης String . Με τη χρήση της while διαβάζουμε γραμμή γραμμή τα περιεχόμενα της reader με τη χρήση της μεθόδου readLine(f.4) και τις τοποθετούμε στη sb με τη χρήση της μεθόδου append(f.4.i). Κλείνουμε το isr(InputStream) με τη χρήση της μεθόδου close(), και τέλος μετατρέπουμε τα περιεχόμενα του string builder sb σε String με τη χρήση της μεθόδου toString() και το τοποθετούμε στη μεταβλητή result(String) . Πάλι σε πρείπτωση αποτυχίας, η catch αναλαμβάνει την καταγραφή του σφάλματος.
Στο τρίτο και πιο ενδιαφέρον σκέλος, έχοντας την απάντηση του server σε string και υπό τη μορφή κωδικοποίησης JSON, αφού ο server εξάγει την απάντηση του σε αυτή τη μορφή, θα πάρουμε μία μία τις τιμές και θα τις εμφανίσουμε στις ανάλογες ετικέτες. Αναλυτικότερα, δηλώνουμε τον πίνακα jarray της κλάσης JSONArray και τον γεμίζουμε με την απάντηση του server(k.1) με τη χρήση της for προσπελαύνουμε όλον τον πίνακα και για κάθε σειρά παίρνουμε το αντικείμενο json της κλάσης JSONObject, που της αντιστοιχεί(k.2.i). Με τη χρήση της μεθόδου getString(**"column name"**) του αντικειμένου json παίρνουμε την τιμή που αντιστοιχεί στη

συγκεκριμένη ταμπέλα. Στην πραγματικότητα παίρνουμε την τιμή που έχει κάθε σειρά στη στήλη που του ζητάμε(**"column name"**).Αυτήν την τιμή την εισάγουμε ως κείμενο στην ανάλογη ετικέτα με τη χρήση της μεθόδου setText("….").

6.5.6 Permissions

Το αρχείο AndroidManifest.xml είναι το αρχείο στο οποίο περιέχονται όλες οι πληροφορίες για την εφαρμογή μας, συμπεριλαμβανομένων και των δικαιωμάτων. Χωρίς αυτό το αρχείο, το πρόγραμμα αδυνατεί να κάνει compile τον κώδικα. Μέσα σε αυτό το αρχείο πρέπει να περιγράψουμε ποια είναι τα δικαιώματα τα οποία πρέπει να έχει η εφαρμογή. Είναι αυτά τα δικαιώματα για τα οποία μας ειδοποιεί το Android Market κάθε φορά που κατεβάζουμε μία εφαρμογή. Στην υφιστάμενη εφαρμογή, χρειαζόμαστε δικαιώματα μόνο για τα SMS και το intenet. Ειδικότερα παρατίθεται οκώδικας του αρχείου.

```
1)  <?xml version="1.0" encoding="utf-8"?>
2)  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
    a)  package="com.project.easygardener"
    b)  android:versionCode="1"
    c)  android:versionName="1.0" >

    d)  <uses-sdk
        i)  android:minSdkVersion="8"
        ii)  android:targetSdkVersion="21" />
    e)  <uses-permission android:name="android.permission.INTERNET"/>
    f)  <uses-permission android:name="android.permission.SEND_SMS"/>

    g)  <application
        i)  android:allowBackup="true"
        ii)  android:icon="@drawable/ic_launcher"
        iii)  android:label="@string/app_name"
        iv)  android:theme="@style/AppTheme" >
        v)  <activity
            (1)  android:name=".MainActivity"
            (2)  android:label="@string/app_name" >
            (3)  <intent-filter>
                (a)  <action android:name="android.intent.action.MAIN" />

                (b)  <category android:name="android.intent.category.LAUNCHER" />
            (4)  </intent-filter>
        vi)  </activity>
    h)  </application>

3)  </manifest>
```

Στις γραμμές 2.e και 2.f καταχωρούμε το δικαίωμα της εφαρμογής να στέλνει SMS(**"android.permission.SEND_SMS"**) και να χρησιμοποιεί το internet(**"android.permission.INTERNET"**). Το σύνολο του υπόλοιπου κώδικα είναι αυτόματα παργόμενο από την πλατφόρμα.

7. ΚΕΦΑΛΑΙΟ 7

ΣΥΝΟΛΙΚΗ ΛΕΙΤΟΥΡΓΙΑ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ

Όσον αφορά τη συνολική λειτουργία του συστήματος στην πράξη, παραθέτουμε παρακάτω δύο επιδεικτικά σενάρια με τα ανάλογα screenshots.

7.1 Σενάριο 1 Χειροκίνητη Έναρξη Ποτίσματος

Στο πρώτο σενάριο ο χρήστης θα χρησιμοποιήσει τη δυνατότητα του συστήματος να ξεκινήσει το πότισμα χειροκίνητα. Αρχικά ευρισκόμενος στην πρώτη οθόνη του συστήματος, μεταφέρεται στο κεντρικό μενού πατώντας το μοναδικό κουμπί που βρίσκεται εκεί(Menu).

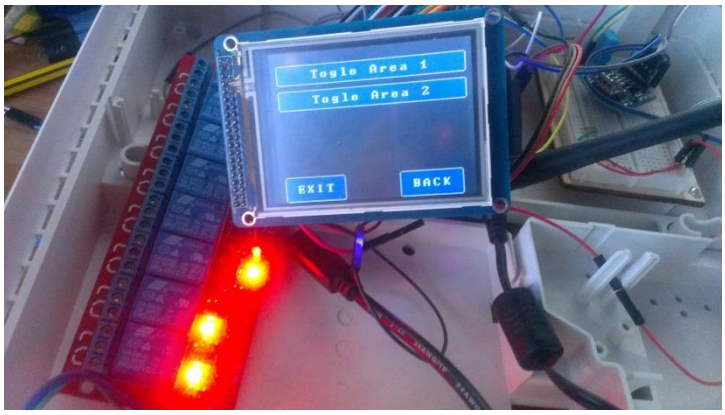
Σχήμα 7-7-1 Κεντρική οθόνη

Αμέσως μεταφέρεται στη δεύτερη οθόνη, όπου μπορεί να επιλέξει να μεταβεί στις ρυθμίσεις του συστήματος(Configuration), ή στον έλεγχο(Control). Επιλέγοντας το κουμπί του ελέγχου μεταφέρεται στην οθόνη ελέγχου των ηλεκτροβανών.

Σχήμα 7-7-2 Main Menu

Σχήμα 7-7-3 Control Screen

Πατώντας το πρώτο ή το δεύτερο κουμπί, ξεκινάμε ή σταματάμε το πότισμα στην εκάστοτε περιοχή. Το σύστημα ανοίγει ή κλείνει την επαφή του ρελέ και το διαπιστώνουμε από τα ενδεικτικά LED λειτουργίας της πλακέτας.



Σχήμα 7-7-4 Control Screen 2Realay Board

Με το κουμπί **Back** μπορεί να μεταφερθεί στην προηγούμενη οθόνη, και με το κουμπί **Exit,** μεταφέρεται αμέσως στην αρχική οθόνη.

7.2 Σενάριο 2 Αλλαγή στο Πρόγραμμα Ποτίσματος

Σε αυτό το σενάριο, ο χρήστης μεταβαίνει στο μενού με τις ρυθμίσεις του συστήματος, όπου θα αλλάξει την καθορισμένη ώρα του πρώτου προγράμματος ποτίσματος.

Αρχικά, από τη δεύτερη οθόνη, επιλέγοντας το κουμπί **Configuration** μεταφέρεται στην οθόνη με τις τρεις επιλογές ρυθμίσεων.

Σχήμα 7-7-5 Ρυθμίσεις

Με το κουμπί **Schedule** μεταφέρεται στην επόμενη οθόνη όπου πρέπει να διαλέξει ποιο από τα δύο προγράμματα ποτίσματος θέλει να αλλάξει.

Σχήμα 7-7-6 Schedule Screen

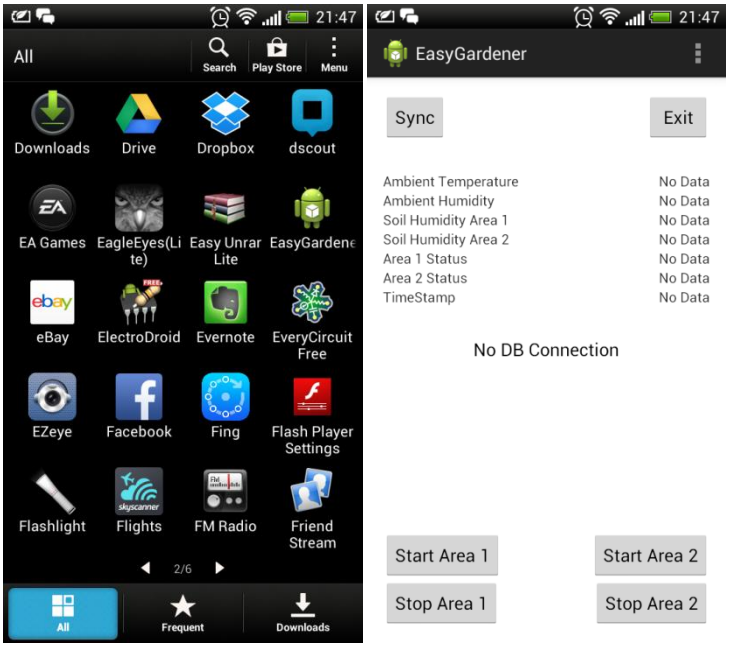
Έπειτα αφού επιλέξει το πρώτο πρόγραμμα(**schedule 1)**, μεταβαίνει στην τελική οθόνη του σεναρίου.

Σχήμα 7-7-7 Edit Schedule 1 screen

Με τα κουμπιά "+" και "-" αυξομειώνει την ώρα και τα λεπτά του προγράμματος. Τέλος, πατώντας το κουμπί **Save,** αποθηκεύει τις αλλαγές που έκανε.

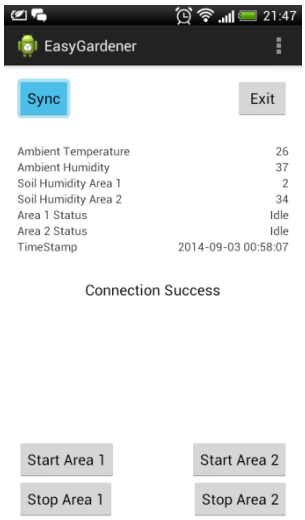
7.3 Σενάριο 3 Απομακρυσμένος έλεγχος

Σε αυτό το σενάριο, ο χρήστης χρησιμοποιεί την android εφαρμογή και ελέγχει το σύστημα απομακρυσμένα. Η εφαρμογή ονομάζεται "easy gardener" .



Σχήμα 7-8 Easy Gardener Interface

Ο χρήστης, πατώντας το κουμπί "Sync" η εφαρμογή διαβάζει τη βάση δεδομένων και εμφανίζει τα στοιχεία στην οθόνη.

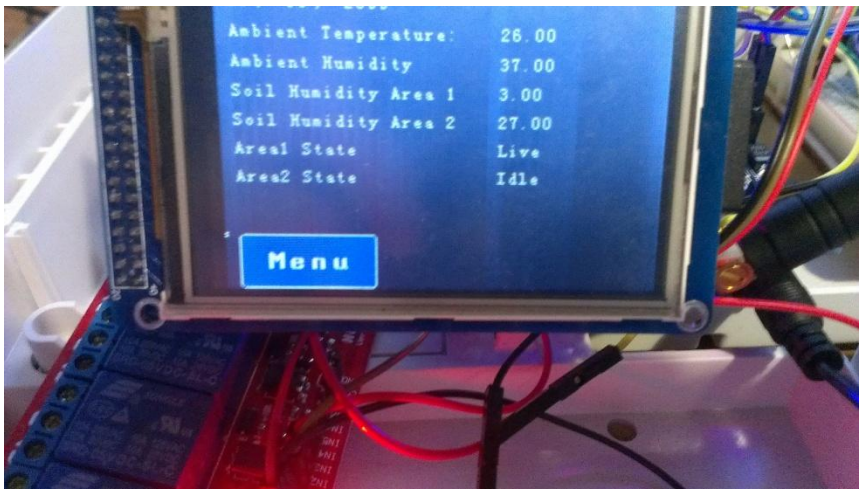


Σχήμα 7-9 Sync Pressed

Διαβάζοντας τις μετρήσεις των αισθητήρων και την κατάσταση του συστήματος, ο χρήστης καταλαβαίνει αν ο χώρος χρειάζεται πότισμα ή αν ποτίζεται ήδη. Έπειτα, αν αποφασίσει να ποτίσει, πατά το κουμπί "Start Area1" και η εφαρμογή με ένα μήνυμα SMS θα ενεργοποιήσει το ανάλογο ρελέ του συστήματος ούτως ώστε να αρχίσει το πότισμα στην ανάλογη περιοχή. Η εφαρμογή μόλις στήλει το SMS ειδοποιεί το χρήστη ότι το μήνυμα εστάλη. Και μετά από λίγο θα αλλάξει και η κατάσταση της εν λόγω περιοχής από "Idle" σε "Live".

Σχήμα 7-10 Send ok Message

Ομοίως, όταν θελήσει να σταματήσει το πότισμα της εν λόγω περιοχής, με το κουμπί "Stop Area 1" πάλι με SMS η εφαρμογή κλείνει το ρελέ του συστήματος. Πάλι, η κατάσταση του χώρου αλλάζει από "Live" σε "Idle".



Σχήμα 7-11 Main Screen Altered

8. ΚΕΦΑΛΑΙΟ 8

ΣΥΜΠΕΡΑΣΜΑΤΑ

Αναφορικά με το παρόν project, πρέπει να αναφερθεί όσον αφορά το σχεδιαμό, ότι έχει επιχειρηθεί μία πρώτου επιπέδου προσέγγιση και η μελλοντική σχεδιαστική του εξέλιξη μπορεί να είναι πολύ μεγάλη, δεδομένου ότι η αυθονία και πικοιλομορφία των υλικών που αφορούν την οικογένεια Arduino είναι τεράστια, πράγμα που σημαίνει ότι για κάθε κομμάτι του συστήματος, μπορούν να υλοποιηθούν πολλές διαφορετικές λύσεις, ανάλογα τις εκάστοτε απαιτήσεις ή τους ανάλογους περιορισμούς. Σίγουρα, η παρούσα υλοποίηση αποτελεί μία σχεδόν ολοκληρωμένη λύση για το πρόβλημα το οποίο καλείται να λύσει. Παρόλα αυτά, δεν παύει να χρήζει εξέλιξης και ανανέωσης.

Πρέπει τέλος να παρατηρήσουμε ότι η ποικιλία και η γκάμα των υλικών που αφορούν την οικογένεια Arduino είναι πάρα πολύ μεγάλη, αλλά στην Ελληνική αγορά οι επιλογές είναι μειωμένες και οικονομικά ασύμφορες σε σύγκριση με την αγορά του εξωτερικού. Η παγκόσμια κοινότητα που ασχολείται με το Arduino είναι πολύ μεγάλη και συνεχώς αυξάνεται. Αναπτύσσονται συνεχώς νέα project και γεννιούνται νέες ιδέες οι οποίες μοιράζονται ανοικτά και δωρεάν στο κοινό και σε όποιον μπορεί να τα αξιοποιήσει. Μέρα με τη μέρα, όλο και πιο πολλές εταιρείες που ασχολούνται με συστήματα αυτοματισμού και ελέγχου συμπεριλαμβάνουν το Arduino και συσκευές συμβατές με αυτό, στον κατάλογό τους (Smart Things). Έχει ξεκινήσει το ενδιαφέρον ολοένα να εντείνεται , όσον αφορά την ανάπτυξη καινούριων ιδεών βασισμένων αποκλειστικά στο Arduino και στην οικογένεια προϊόντων συμβατών με αυτό. Σίγουρα, και το Arduino αλλά και η κοινότητα, βρίσκονται στα πρώτα τους βήματα ακόμα. Όμως επειδή η εξέλιξη δεν σταματά ποτέ, σύντομα το Arduino θα αποτελεί ένα μεγάλο κεφάλαιο, προσφιλές και προσοδοφόρο, στην αγορά και στον κόσμο των προγραμματιστών.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Julian W. Garder, (2000), Μικροαισθητήρες. Θεσσαλονίκη: Εκδόσεις Τζιόλα.
2. Αισθητήρες Ημιαγωγών, Αισθητήρες θερμικοί, μηχανικοί, μαγνητικοί, αισθητήρες ακτινοβολίας και χημικοί αισθητήρες. (Πάτρα 2004), Αθανάσιος Α. Αργυρίου Επίκουρος Καθηγητής Πανεπιστήμιο Πατρών - Τμήμα Φυσικής
3. Steven F.Barret-Daniel J. Pack, Microcontrollers Fundamentals for engineers and scientists
4. John G.Webster Measurement, Instrumentation
5. Waldemar Nawrocki Measurement Systems And Sensors
6. Michael Margolis, Arduino CookBook
7. Ed Burnete, Hello Android
8. Βικιπαίδια. Online Εγκυκλοπαίδεια. http://en.wikipedia.org/wiki/Microcontroller
9. www.atmel.com
10. www.arduino.cc
11. www.raspberrypi.org
12. www.henningkarlsen.com/ Παραγωγή και Διάθεση βιβλιοθηκών Arduino .

ΠΑΡΑΡΤΗΜΑ

```
1)  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    a)  xmlns:tools="http://schemas.android.com/tools"
    b)  android:layout_width="match_parent"
    c)  android:layout_height="match_parent"
    d)  android:background="@color/white"
    e)  android:paddingBottom="@dimen/activity_vertical_margin"
    f)  android:paddingLeft="@dimen/activity_horizontal_margin"
    g)  android:paddingRight="@dimen/activity_horizontal_margin"
    h)  android:paddingTop="@dimen/activity_vertical_margin"
    i)  tools:context="com.project.easygardener.MainActivity" >

    j)  <Button
        i)   android:id="@+id/bsync"
        ii)  android:layout_width="wrap_content"
        iii) android:layout_height="wrap_content"
        iv)  android:layout_alignParentLeft="true"
        v)   android:layout_alignParentTop="true"
        vi)  android:text="Sync"
        vii) tools:ignore="HardcodedText" />

    k)  <Button
        i)   android:id="@+id/bexit"
        ii)  android:layout_width="wrap_content"
        iii) android:layout_height="wrap_content"
        iv)  android:layout_alignParentRight="true"
        v)   android:layout_alignParentTop="true"
        vi)  android:text="Exit"
        vii) tools:ignore="HardcodedText"/>

    l)  <TextView
        i)   android:id="@+id/textView2"
        ii)  android:layout_width="wrap_content"
        iii) android:layout_height="wrap_content"
        iv)  android:layout_alignParentLeft="true"
        v)   android:layout_below="@+id/textView1"
        vi)  android:text="Ambient Humidity"
        vii) tools:ignore="HardcodedText" />

    m)  <TextView
        i)   android:id="@+id/textView3"
        ii)  android:layout_width="wrap_content"
        iii) android:layout_height="wrap_content"
        iv)  android:layout_alignParentLeft="true"
        v)   android:layout_below="@+id/textView2"
        vi)  android:text="Soil Humidity Area 1"
        vii) tools:ignore="HardcodedText" />

    n)  <TextView
        i)   android:id="@+id/textView4"
        ii)  android:layout_width="wrap_content"
        iii) android:layout_height="wrap_content"
        iv)  android:layout_alignParentLeft="true"
        v)   android:layout_below="@+id/textView3"
        vi)  android:text="Soil Humidity Area 2"
        vii) tools:ignore="HardcodedText"/>

    o)  <TextView
        i)   android:id="@+id/textView5"
        ii)  android:layout_width="wrap_content"
        iii) android:layout_height="wrap_content"
        iv)  android:layout_alignParentLeft="true"
        v)   android:layout_below="@+id/textView4"
        vi)  android:text="Area 1 Status"
        vii) tools:ignore="HardcodedText" />

    p)  <TextView
        i)   android:id="@+id/textView6"
        ii)  android:layout_width="wrap_content"
        iii) android:layout_height="wrap_content"
        iv)  android:layout_alignParentLeft="true"
        v)   android:layout_below="@+id/textView5"
        vi)  android:text="Area 2 Status"
        vii) tools:ignore="HardcodedText"/>
```

q) <TextView
i) android:id="@+id/tvah"
ii) android:gravity="right"
iii) android:layout_width="wrap_content"
iv) android:layout_height="wrap_content"
v) android:layout_alignLeft="@+id/tvsa1"
vi) android:layout_alignParentRight="true"
vii) android:layout_below="@+id/tvat"
viii) android:text="No Data"
ix) tools:ignore="HardcodedText" />

r) <TextView
i) android:id="@+id/tvsa1"
ii) android:gravity="right"
iii) android:layout_width="wrap_content"
iv) android:layout_height="wrap_content"
v) android:layout_alignBaseline="@+id/textView3"
vi) android:layout_alignBottom="@+id/textView3"
vii) android:layout_alignLeft="@+id/tvas1"
viii) android:layout_alignParentRight="true"
ix) android:text="No Data"
x) tools:ignore="HardcodedText" />

s) <TextView
i) android:id="@+id/tvsa2"
ii) android:gravity="right"
iii) android:layout_width="wrap_content"
iv) android:layout_height="wrap_content"
v) android:layout_alignBaseline="@+id/textView4"
vi) android:layout_alignBottom="@+id/textView4"
vii) android:layout_alignLeft="@+id/tvas1"
viii) android:layout_alignRight="@+id/tvsa1"
ix) android:text="No Data"
x) tools:ignore="HardcodedText" />

t) <TextView
i) android:id="@+id/tvas1"
ii) android:gravity="right"
iii) android:layout_width="wrap_content"
iv) android:layout_height="wrap_content"
v) android:layout_alignParentRight="true"
vi) android:layout_below="@+id/tvsa2"
vii) android:layout_toRightOf="@+id/textView1"
viii) android:text="No Data"
ix) tools:ignore="HardcodedText" />

u) <TextView
i) android:id="@+id/tvas2"
ii) android:layout_width="wrap_content"
iii) android:layout_height="wrap_content"
iv) android:layout_alignBaseline="@+id/textView6"
v) android:layout_alignBottom="@+id/textView6"
vi) android:layout_alignRight="@+id/tvas1"
vii) android:layout_toRightOf="@+id/textView1"
viii) android:gravity="right"
ix) android:text="No Data"
x) tools:ignore="HardcodedText" />

v) <Button
i) android:id="@+id/bstop1"
ii) android:layout_width="wrap_content"
iii) android:layout_height="wrap_content"
iv) android:layout_alignParentBottom="true"
v) android:layout_alignParentLeft="true"
vi) android:text="Stop Area 1"
vii) tools:ignore="HardcodedText" />

w) <Button
i) android:id="@+id/bstop2"
ii) android:layout_width="wrap_content"
iii) android:layout_height="wrap_content"
iv) android:layout_alignParentBottom="true"
v) android:layout_alignParentRight="true"
vi) android:text="Stop Area 2"
vii) tools:ignore="HardcodedText" />

x) <Button
i) android:id="@+id/bstart1"
ii) android:layout_width="wrap_content"
iii) android:layout_height="wrap_content"

iv) android:layout_above="@+id/bstop1"
v) android:layout_alignParentLeft="true"
vi) android:text="Start Area 1"
vii) tools:ignore="HardcodedText"/>

y) <Button
i) android:id="@+id/bstart2"
ii) android:layout_width="wrap_content"
iii) android:layout_height="wrap_content"
iv) android:layout_above="@+id/bstop2"
v) android:layout_alignParentRight="true"
vi) android:text="Start Area 2"
vii) tools:ignore="HardcodedText"/>

z) <TextView
i) android:id="@+id/tvconnect"
ii) android:layout_width="wrap_content"
iii) android:layout_height="wrap_content"
iv) android:layout_below="@+id/textView6"
v) android:layout_centerHorizontal="true"
vi) android:layout_marginTop="50dp"
vii) android:text="No DB Connection"
viii) android:textAppearance="?android:attr/textAppearanceMedium"
ix) tools:ignore="HardcodedText" />

aa) <TextView
i) android:id="@+id/tvtimestamp"
ii) android:layout_width="wrap_content"
iii) android:layout_height="wrap_content"
iv) android:layout_alignLeft="@+id/textView6"
v) android:layout_below="@+id/textView6"
vi) android:text="TimeStamp"
vii) tools:ignore="HardcodedText"/>

bb) <TextView
i) android:id="@+id/tvts"
ii) android:layout_width="wrap_content"
iii) android:layout_height="wrap_content"
iv) android:layout_alignBaseline="@+id/tvtimestamp"
v) android:layout_alignBottom="@+id/tvtimestamp"
vi) android:layout_alignRight="@+id/tvas2"
vii) android:layout_toRightOf="@+id/textView1"
viii) android:gravity="right"
ix) android:text="No Data"
x) tools:ignore="HardcodedText" />

cc) <TextView
i) android:id="@+id/tvat"
ii) android:gravity="right"
iii) android:layout_width="wrap_content"
iv) android:layout_height="wrap_content"
v) android:layout_above="@+id/textView2"
vi) android:layout_alignLeft="@+id/tvah"
vii) android:layout_alignRight="@+id/tvah"
viii) android:text="No Data"
ix) tools:ignore="HardcodedText" />

dd) <TextView
i) android:id="@+id/textView1"
ii) android:layout_width="wrap_content"
iii) android:layout_height="wrap_content"
iv) android:layout_alignLeft="@+id/textView2"
v) android:layout_below="@+id/bsync"
vi) android:layout_marginTop="37dp"
vii) android:text="Ambient Temperature"
viii) tools:ignore="HardcodedText" />

2) </RelativeLayout>