ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΙΑΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Μελέτη & Κατασκευή ψηφιακού θερμομέτρου με μικροελεγκτή

ΕΙΣΗΓΗΤΗΣ :
ΜΗΝΑΔΑΚΗΣ ΙΩΑΝΝΗΣ

ΣΠΟΥΔΑΣΤΗΣ :
ΚΟΡΝΕΛΑΚΗΣ ΧΑΡΑΛΑΜΠΟΣ

ΗΡΑΚΛΕΙΟ 2008

ΠΕΡΙΕΧΟΜΕΝΑ

ΑΝΤΙΚΕΙΜΕΝΟ ΤΗΣ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ ..... - 4 -

ΚΕΦΑΛΑΙΟ 1 ..... - 5 -

1.1. Τί είναι θερμοκρασία ..... - 5 -

1.2. Τί είναι το θερμόμετρο ..... - 6 -

1.3. Η εξέλιξη των θερμομέτρων και των θερμοκρασιακών κλιμάκων ..... - 7 -

ΚΕΦΑΛΑΙΟ 2 ..... - 12 -

2.1. Εισαγωγή στους αισθητήρες θερμοκρασίας. ..... - 12 -

2.2.Πρακτικά Αισθητήρια Θερμοκρασίας. ..... - 12 -

2.2.1. LM34CZ / LM35CZ ..... - 13 -

2.2.2. PT100 ..... - 13 -

ΚΕΦΑΛΑΙΟ 3 ..... - 14 -

3.1. Η ακρίβεια ..... - 14 -

3.2. Resolution (Διακριτικότητα ή ανάλυση) ..... - 14 -

3.3. Υστέρηση ..... - 17 -

3.4. Ορολογία Για Αισθητήρες ..... - 18 -

Κεφάλαιο 4 ..... - 22 -

4.1. Γενική Περιγραφή της συσκευής ..... - 22 -

4.2. Ο ATMEL mega 128 ..... - 23 -

4.3. USART (Σειριακός Προγραμματισμός) ..... - 24 -

4.4. USART Register Description ..... - 25 -

4.4.1. USARTn I/O Data Register-UDRn ..... - 25 -

4.4.2. USART Control And Sta tus Register A-UCSRnA ..... - 26 -

4.4.3. USARTn Control and Status Register B – UCSRnB ..... - 28 -

4.4.4. USARTn Control and Status Register C – UCSRnC ..... - 29 -

4.4.5. USART Baud Rate Registers - UBRRnL and UBRRnH ..... - 32 -

4.5. ADC (Analog to Digital Converter) ..... - 33 -

4.6. ADC Register Description ..... - 33 -

4.6.1. ADC Multiplexer Selection Register – ADMUX ..... - 33 -

4.6.2. ADC Control and Status Register A – ADCSRA ..... - 34 -

4.6.3 The ADC Data Register – ADCL and ADCH ..... - 36 -

4.7. Οθόνη Υγρών Κρυστάλλων ..... - 37 -

4.8. Αισθητήρια Θερμοκρασίας AD22100A ..... - 39 -

4.8.1. MICROPROCESSOR A/D INTERFACE ISSUES ..... - 41 -

4.8.2. RATIOMETRICITY CONSIDERATIONS ..... - 41 -

4.9 USB Flash Device (VDRIVE) ..... - 43 -

Κεφάλαιο 5 ..... - 46 -

5.1. Χρήση Συσκευής και Εντολές Λειτουργίας ..... - 46 -

5.1.1. Έναρξη Λειτουργίας ..... - 46 -

5.1.2. Επιλογή 1η – Values ..... - 47 -

5.1.3. Επιλογή 2η – Range ..... - 47 -

5.1.4. Επιλογή 3η – Info ..... - 47 -

5.1.5. Επιλογή 4η – Time ..... - 48 -

5.1.6. Επιλογή 5η – Date ..... - 48 -

5.1.7. Επιλογή 6η – Reset ..... - 49 -

5.1.8. Επιλογή 7η – Start ..... - 49 -

5.2. Βασική Λειτουργία της Συσκευής ..... - 50 -

Κεφάλαιο 6 ..... - 53 -

6.1.Schematics και Block Diagram ..... - 53 -

ΠΑΡΑΡΤΗΜΑ ..... **- 59 -**

ΑΝΤΙΚΕΙΜΕΝΟ ΤΗΣ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Η ανάγκη για μαζική αποθήκευση μετρήσεων σε κάποιο αποθηκευτικό μέσο, ήταν η αφορμή για αυτήν την πτυχιακή εργασία. Αντικείμενο της πτυχιακής εργασίας είναι η σχεδίαση και η κατασκευή ενός συστήματος το οποίο θα εξυπηρετεί την λήψη και την αποθήκευση μετρήσεων σε μια συσκευή USB Flash.

Οι USB Flash είναι αποθυκευτικά μέσα, τα οποία μπορούν να αποθηκεύσουν μεγάλο όγκο δεδομένων, διατηρώντας το περιεχομενό τους για μεγάλο χρονικό διάστημα.Η λειτουργία του συστήματος βασίζεται σε έναν μικροεπεξεργαστή, ο οποίος είναι υπεύθυνος για την λήψη των μετρήσεων από 4 Εισόδους (Αισθητήρια) και στην συνέχεια την αποθηκευσή τους , στο USB Flash. Οι μετρήσεις που πραγματοποιεί το σύστημα , είναι μετρ ήσεις θερμοκρασίας (συνεχής τάση από 0 Volts έως 5 Volts).

Για την απεικόνιση των λειτουργιών του συστήματος χρησιμοποιήθηκε μία οθόνη υγρών κρυστάλλων 20x4 (Liquid Crystal Display), ενώ για την επιλογή των εντολών στο Menu χρησιμοποιήθηκε ένα πληκτρολόγ ιο 4x4 (16 πλήκτρων). Το σύστημα διαθέτει ISP (In System Programming), δίνοντας την δυνατότητα για άμεσο προγραμματισμό του μικροεπεξεργαστή χωρίς να απαιτείται η απομάκρυνσή του από το σύστημα.

Η κατασκευή αποτελείται από μία ηλ εκτρονική πλακέτα. Η πλακέτα περιλαμβάνει τον μικροεπεξεργαστή ATMEL mega 128 με το κύκλωμα του κρυσταλλικού ταλαντωτή, το βύσμα τροφοδοσίας του συστήματος με το κύκλωμα σταθεροποίησης της τάσεως , τον συνδετήρα ISP προγραμματισμού, τα 4 Κανάλια/Εισόδους (Αισθητήρια) των μετρήσεων και τα απαραίτητα βύσματα για την σύνδεση στην πλακέτα της οθόνης υγρών κρυστάλλων (LCD) και του πληκτρολογίου.

ΚΕΦΑΛΑΙΟ 1

1.1. Τί είναι θερμοκρασία

Με ποιοτικό τρόπο, μπορούμε να περιγράψουμε την θερμοκρασία ενός αντικειμένου, ως αυτό που προσδιορίζει την αίσθηση της ζέστης ή του ψυχρού κατα την επαφή μας με αυτό.

Είναι γνωστό ότι, όταν δύο αντικείμενα,τοποθετηθούν μαζί (Θερμική Επαφή), το αντικείμενο με την υψηλότερη θερμοκρασία τείνει να ψυχραίνεται, ενώ το ποιό ψυχρό αντικείμενο τείνει να ζεσταίνεται. Η διαδικασία αυτή συνεχίζεται μέχρι να φτάσουν σε ένα σημείο όπου δεν υπάρχει αλλαγή στις αισθήσεις μας. Όταν οι θερμικές αλλαγές, ανάμεσα στα δύο αντικείμενα (Συστήματα) που έχουν την θερμική επαφή σταματήσουν, τότε μπορούμε να ορίσουμε τη θερμοκρασία του συστήματος, λέγοντας ότι η θερμοκρασία είναι μια μακροσκοπική παράμετρος η οποία έχει την ιδιότητα να είναι ίδια για δυο συστήματα τα οποία βρίσκονται σε θερμική ισορροπία.

Αν πειραματιστούμε με περισσότερα από δυο συστήματα, μπορούμε να διαπιστώσουμε, ότι όταν αυτά έρθουν σε θερμική ισορροπία, η θερμική ισορροπία δεν εξαρτάται από την φύση του αντικειμένου που χρησιμοποιείται. Αλλά ειδικότερα : *Αν δυο ξεχωριστά συστήματα είναι σε θερμική ισορροπία με ένα τρίτο, τότε πρέπει και αυτα να 'ναι σε θερμική ισορροπία μεταξύ τους ,* και όλα έχουν την ίδια θερμοκρασία ανεξάρτητα από το είδος του συστήματος.

Η διατύπωση με τα πλάγια γράμματα μπορεί να επαναδιατυπωθεί ως εξής : *Αν τρία ή περισσότερα συστήματα είναι σε θερμική ισορροπία μεταξύ τους, τότε οποιαδήποτε δυο από αυτά , είναι σε θερμική ισορροπία μεταξύ τους.*

Ένα από αυτά τα τρία συστήματα μπορεί να είναι ένας αισθητήρας βαθμονομημένος να μετρά την θερμοκρασία. Αυτό είναι το θερμόμετρο. Όταν ένα βαθμονομημένο θερμόμετρο τοποθετηθεί σε θερμική επαφή με το σύστημα και φτάσουν σε θερμική ισορροπία, τότε έχουμε μια ποσοτική μέτρηση της θερμοκρασίας του συστήματος. Για παράδειγμα το υδραργιρικό θερμόμετρο τοποθετείται κάτω από την γλώσσα του ασθενούς και αφήνεται να φτάσει σε θερμική ισορροπία με το στόμα του ασθενούς.Τότε βλέπουμε κατα πόσο έχει διασταλεί ο υδράργυρος και διαβάζουμε στην κλίμακα του θερμομέτρου για να δούμε την θερμοκρασία του ασθενούς.

1.2. Τί είναι το θερμόμετρο

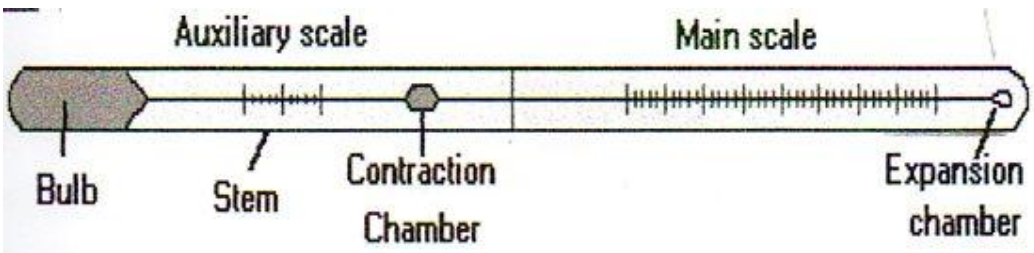
Το θερμόμετρο είναι ένα όργανο που μετράει την θερμοκρασία ενός συστήματος με έναν ποσοτικό τρόπο. Ο ποιό εύκολος τρόπος για να γίνει αυτό είναι να βρεθεί μια ουσία που έχει την ιδιότητα "να αλλάζει" με έναν ομαλό τρόπο σε σχέση με την θερμοκρασία. Ο πιο «ομαλός» τρόπος είναι ο γραμμικός :

$$T(x) = ax + b$$

όπου : **T** ειναι η θερμοκρασία της ουσίας που μετατρέπεται καθώς η ιδιότητα **x** της ουσίας αλλάζει ενώ **a** και **b** είναι σταθερές που εξαρτώνται από την ουσία που χρησιμοποιείτε και μπορεί να εξισωθεί ορίζοντας δυο σημεία θερμοκρασίας πάνω στην κλίμακα όπως 32^{o} Kelvin ,όπου το νερό παγώνει και 212^{o} K για το σημείο βρασμού.

Για παράδειγμα, το στοιχείο υδράργυρος είναι ένα υγρό με μια έκταση θερμοκρασίας -38,9 ^{o}C μέχρι 356,7 ^{o}C (παρακάτω θα υπάρξει αναφορά για τους βαθμούς Κελσίου ^{o}C).

Σαν υγρό μέταλλο ο υδράργυρος διαστέλλεται καθώς ζεσταίνεται. Ο ρυθμός διαστολής είναι γραμμικός και μπορεί να βαθμονομηθεί με ακρίβεια.



Σχήμα 1.1. *Υδραργυρικό θερμόμετρο*

Το υδραργυρικό θερμόμετρο στο παραπάνω σχήμα περιέχει υδράργυρο. Με αυτό τον τρόπο του δίνεται η δυνατότητα να διαστέλλεται μέσα σε ένα τριχοειδή σωλήνα.Ο ρυθμός διαστολής είναι ρυθμισμένος στη γυάλινη κλίμακα.

1.3. Η εξέλιξη των θερμομέτρων και των θερμοκρασιακών κλιμάκων

Μια από τις πρώτες προσπάθειες για να φτιαχτεί μια πρότυπη κλίμακα θερμοκρασίας ήταν το 170 μ.Χ., όταν ο Galen , στα ιατρικά του συγγράμματα, πρότεινε μια πρότυπη «ουδέτερη» θερμοκρασία φτιαγμένη από ίσες ποσότητες βρασμένου νερού και πάγου. Και στις δυο άκρες αυτής της κλίμακας θερμοκρασίας υπάρχουν τέσσερις βαθμοί ζέστης και τέσσερις βαθμοί κρύου.

Οι πρώτες συσκευές που χρησιμοποιήθηκαν για την θερμοκρασία λεγόταν *θερμοσκόπια*.

Σχήμα 1.2. *Θερμοσκόπιο*

Το θερμοσκόπιο αποτελείτε από ένα γυάλινο βολβό με ένα μακρύ σωλήνα, επεκτεινόμενο προς τα κάτω που καταλήγει σε ένα δοχε ίο με χρωματισμένο νερό, αν και ο Galileo το 1610 μ.Χ. λέγεται ότι χρησιμοποίησε κρασί. Κάποια ποσότητα του αέρα αφαιρέθηκε πριν τοποθετηθεί μέσα το υγρό, προκαλώντας το υγρό να ανέβει μέσα στο σωλήνα. Έτσι ώστε ο αέρας που έμεινε στο βολβό, αν ζεσταινότα ν ή πάγωνε, η στάθμη του υγρού μέσα στο σωληνάριο να άλλαζε αντικατοπτρίζοντας την αλλαγή στην θερμοκρασία

του αέρα. Μια χαραγμένη κλίμακα στον σωλήνα επέτρεπε την ποσοτική μέτρηση των διακυμάνσεων.

Ο αέρας μέσα στο βολβό αναφερόταν ως το θερμομετρικό όργα νο, του οποίου οι ιδιότητες αλλάζουν ανάλογα με τη θερμοκρασία.

Το 1641 μ.Χ. φτιάχτηκε το πρώτο σφραγισμένο θερμόμετρο που χρησιμοποιούσε υγρό αντί του αέρα σαν θερμομετρικό όργανο. Έγινε για τον Ferdinand II, Δούκα της Τοσκάνης. Το θερμόμετρο, χρησιμοποιο ύσε μια σφραγισμένη αλκοολούχα συσκευή, με 50 «βαθμούς» σημειωμένες στο σωλήνα του,χωρίς κάποιο «κανονισμένο σημείο» για να μηδενίζει την κλίμακα.Τα θερμόμετρα αυτά λεγόταν θερμόμετρα οινοπνεύματος.

Ο Robert Hook έμπορος, το 1664 μ.Χ. χρησιμοποίησε μια κόκ κινη βαφή στο οινόπνευμα. Η κλίμακα του, που για κάθε βαθμό απεικόνιζε αύξηση του όγκου ίση με το 1/500 του όγκου του θερμομετρικού υγρού χρειάζονταν ένα μόνο σημείο κανονισμένο. Επέλεξε το σημείο όπου το νερό παγώνει. Βαθμολογόντας το έτσι ο Hook έδειξε ότι μια πρότυπη (μοντέλο) κλίμακα θα μπορούσε να καθιερωθεί για θερμόμετρα που χαρακτηρίζονται από ποικιλία μεγεθών. Το πρότυπο θερμόμετρο του Hook έγινε γνωστό ως το μοντέλο του Κολεγίου του Gresham και το χρησιμοποιούσε ο Βασιλικός Σύλλογος μέχρι το 1709 μ.Χ. Οι πρώτες μετεωρολογικές μετρήσεις που χαρακτηρίζονται από λογικό ειρμό,χρησιμοποίησαν αυτή την κλίμακα .Το 1702 μ.Χ., ο αστρονόμος Ole Roemer από την Κοπεγχάγη, στήριξε την κλίμακα του επάνω σε δυο κανονισμένα σημεία : το χιόνι ή τριμμένο πάγο και το βρασμένο νερό, και κατέγραψε ημερησίως τις θερμοκρασίες στην Κοπεγχάγη το 1708 -1709 μ.Χ. με αυτό το θερμόμετρο.

Το 1724 μ.Χ. ο Gabriel Fahrenheit, κατασκευαστής οργάνων της Daanzing & Amsterdam, χρησιμοποίησε τον υδράργυρο σαν θερμομετρικό υγρό. Η θερμική διαστολή του υδραργύρου ήταν μεγάλη και όμοια, δεν «κολλούσε» στο γυαλί και παρέμενε σε υγρή μορφή σε μεγάλο εύρος θερμοκρασίας. Το ασημί του χρώμα το διευκόλυνε στο διάβασμα.

Ο Fahrenheit περιγράφει πως ρύθμισε την κλίμακα του για το υδραργυρικό θερμόμετρο :

« *Όταν τοποθετούσε το θερμόμετρο σε μείγμα από θαλλασινό νερό ή αλατούχα αμμωνία, πάγο και νερό βρισκόταν ένα σημείο στην κλίμακα που έδειχνε το μηδέν. Ένα δεύτερο σημείο παρατήρησε όταν χρησιμοποίησε το μείγμα χωρίς το αλάτι. Αυτό το σημείο σημειώνοντα ν έως το 30 . Ενώ ένα τρίτο σημείο, το 96 βρισκόταν αν το θερμόμετρο τοποθετηθεί στο στόμα ενός υγιή άντρα και μετρηθεί η θερμοκρασία.»*

- D.G Fahrenheit (London) 33,78,1724

Σε αυτήν την κλίμακα ο Fahrenheit μέτρησε το σημείο βρασμού του νερού στο 212. Αργότερα ρύθμισε το σημείο παγώματος του νερού στο 32, ώστε το διάστημα μεταξύ του σημείου βρασμού και παγώματος του νερού, να αντιπροσωπεύεται από ένα λογικό αριθμό το 180. Οι θερμοκρασίες που μετριούνται σε αυτήν την κλίμακα ορίζονται ως Βαθμοί Fahrenheit - ^{o}F.

Το 1745 μ.Χ., ο Carolus Linnaeus από την Upsula της Σουηδίας, περιέγραψε μια κλίμακα στην οποία το σημείο παγώματος του νερού είναι το μηδέν και το σημείο βρασμού το 100, κάνοντας το θερμόμετρο εκατοντάβαθμο.

Ο Andres Celsius (1701-1744) χρησιμοποίησε την ανάστροφη κλίμακα όπου το 100 ήταν το σημείο παγώματος και το 0 ήταν το σημείο βρασμού.Πάλι το θερμόμετρο ήταν εκατοντάβαθμο.

Ο Sir Hamphrey Davy ήταν ένας λαμπρός επιστήμονας που έκανε σημαντικές ανακαλύψεις : α) οξείδιο του νατρίου (αναισθητικό), β) την ανακάλυψη των στοιχείων σόδα,ποτάσα και βόριο, γ) ηλεκτροσυγκόλληση και την λάμπα ασφαλείας ορυχείων.

Το 1799 έλιωσε δυο πάγους τρίβοντας τους μεταξύ τους.Αυτή η ενέργεια απέδειξε ότι, η ζέστη είναι μορφή ενέργειας.Μέχρι τότε η ζέστη θεωρούνταν ως υγρό μικρού βάρους που λεγόταν **Caloric**. Αυτή η ανακάλυψη άλλαξε τον τρόπο έρευνας της θερμοκρασίας και άνοιξε τον δρόμο για τα εξελιγμένα συστήματα μέτρησης της θερμοκρασίας που μέχρι τότε ήταν τα απλά θερμόμετρα.

Το 1948 μ.Χ. η χρήση του εκατοντάβαθμου εγκαταλείφτηκε, χάρη σε μια νέα κλίμακα που χρησιμοποίησε βαθμούς Celsius - ^{o}C. Η κλίμακα Celsius ορίζεται από μέρη.

i) Το τριπλό σημείο του νερού ορίζεται ως 0,01 ^{o}C.

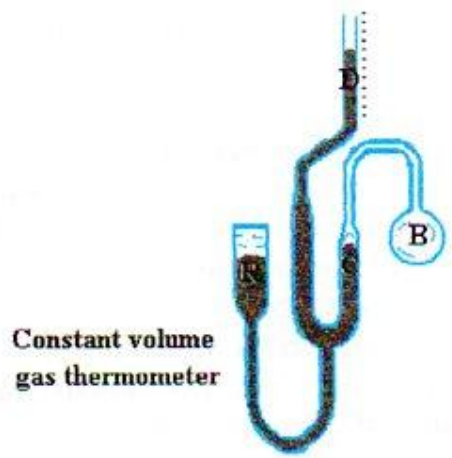
ii) Ένας βαθμός Κελσίου αντιστοιχεί με την αλλαγή της θερμοκρασίας, όσο ενός βαθμού στην κλίμακα του ιδανικού αερίου.

Στην κλίμακα Celsius το σημείο βρασμού του νερού, με σταθερή ατμοσφαιρική πίεση, είναι το 99,975 ^{o}C αντιθέτως με το 100 του εκατοντάβαθμου.

Για την μετατροπή από Celsius σε Fahrenheit πολλαπλασιάζουμε με 1,8 και προσθέτουμε το 32

$$^{o}F = 1{,}8\,^{o}C + 32$$

Το 1780 μ.Χ. ο J.A.C. Charles Γάλλος φυσικός, έδειξε ότι για τις ίδιες αυξήσεις της θερμοκρασίας,όλα τα αέρια παρουσίασαν την ίδια αύξηση του όγκου τους.Επειδή ο συντελεστής διαστολής των αερίων είναι σχεδόν ίδιος, είναι δυνατόν να καθιερωθεί μια θερμοκρασιακή κλίμακα βασισμένη σε ένα σημείο αντί σε δυο, όπως του Fahrenheit και του Celsius. Αυτό μας οδηγεί πίσω στα θερμόμετρα που χρησιμοποιούν το αέριο ως θερμομετρικό μέσο.



Σχήμα 1.3. *Θερμόμετρο σταθερού αερίου*

Σε ένα θερμόμετρο σταθερού όγκου ένας βολβός B , π.χ. υδρογόνου, κάτω απο μια ορισμένη πίεση συνδέεται με ενα μανόμετρο υδραργύρου με σωλήνα μικρού όγκου. Ο βολβός B είναι ο αισθητήρας θερμοκρασίας και πρέπει να περιέχει σχεδόν όλο το υδρογόνο. Η στάθμη του υδρογόνου C μπορεί να ρυθμιστεί ανεβοκατεβάζοντας το απόθεμα στο ντεπόζιτο R. Η πίεση του υδρογόνου που είναι το x στον γραμμικό τύπο είναι η διαφορά μεταξύ των στάθμεων D και C συν της πίεσης άνω του D.

Ο P.Chappius το 1887 μ.Χ. διενήργησε έρευνες στα θερμόμετρα αερίου με σταθερή πίεση ή σταθερό όγκο χρησιμοποιώντας υδρογόνο,άζωτο και διοξείδιο του άνθρακα ως θερμομετρικό μέσο. Βάση των αποτελεσμάτων , η Comite International des Poids et Mesures υιοθέτησε την κλίμακα σταθερού όγκου υδρογόνου βασισμένο στα σημεία παγώματος 0 ^{o}C και βρασμού 100 ^{o}C ως την πρακτική κλίμακα για την διεθνή μετεωρολογία.

Τα πειράματα με θερμόμετρα αερίου έχουν δείξει μικρές διαφορές στην κλίμακα για διάφορα αέρια. Συνεπώς είναι δυνατόν να οριστεί μια κλίμακα

ανεξάρτητου του θερμοκρασιακού μέσου, εάν είναι αέριο χαμηλής πίεσης. Σε αυτήν την περίπτωση, όλα τα αέρια συμπεριφέρονται σαν ιδανικά αέρια και έχουν μια απλή σχέση μεταξύ της πίεσης , όγκου και θερμοκρασίας :

PV = (σταθερά) T

Αυτή η θερμοκρασία ονομάζετε θερμοδυναμική και είναι τώρα αποδεκτή ως η θεμελιώδης μέτρηση της θερμοκρασίας. Σημειώνεται ότι υπάρχει ένα φυσικό μηδέν στην κλίμακα αυτή, στο οποίο η πίεση ενός ιδανικού αερίου είναι μηδέν, μηδενίζοντας επίσης την θερμοκρασία.Παίρνοντας αυτό το σημείο χρειάζεται ακόμα ένα σημείο να οριστεί.

Το έτος 1993 η International Committee of Weights and Measures υιοθέτησε σαν σημείο, το τριπλό σημείο του νερού, στην θερμοκρασία στην οποία το νερό, ο πάγος και οι υδρατμοί συνυπάρχουν σε ισορροπία, η τιμή του οποίου είναι 273,16. Η μονάδα θερμοκρασίας ονομάζεται Kelvin, από τον Lord Kelvin (William Thompson), 1824-1907 και συμβολίζεται με το **K**. Για την μετατροπή απο Κελσίου σε Κέλβιν,προσθέτουμε 273.

Το 1871 ο Sir William Siemens πρότεινε ένα θερμόμετρο του οποίου το θερμομετρικό μέσο,είναι ένας μεταλλικός αγωγός και η αγωγιμότητά του αλλάζει αναλόγως της θερμοκρασίας. Το στοιχείο του λευκόχρυσου δεν οξειδώνεται σε υψηλές θερμοκρασίες και έχει σχετικά ομοιόμορφη μεταβολή στην αντίστασή του, σε μεγάλη κλίμακα αλλαγής θερμοκρασίας. Το θερμόμετρο λευκόχρυσης αντίστασης, χρησιμοποιείτε ευρέως ως θερμο - ηλεκτρικό θερμόμετρο και καλύπτει θερμοκρασίες από -260 έως 1235 ^{o}C.

Το 1826 ο T.J.Seebeck ανακάλυψε ότι, όταν σύρματα διαφορετικών μετάλλων, είναι ενωμένα και θερμαινόμενα, τότε ένα ρεύμα ρέει από το ένα στο άλλο.

Η Η.Ε.Δ που παράγεται, μπορεί να συσχετισθεί με την θερμοκρασία, με αποτέλεσμα, το σύστημα να μπορεί να χρησιμοποιηθεί ως θερμόμετρο, γνωστό ως θερμοζεύγος.Το θερμοζεύγος χρησιμοποιείται στην βιομηχανία και χρησιμοποιούνται διάφορα μέταλλα, όπως λευκόχρυσος, ρόδιο, χρωμιονικελίνη και αλουμίνιο.

ΚΕΦΑΛΑΙΟ 2

2.1. Εισαγωγή στους αισθητήρες θερμοκρασίας .

Η θερμοκρασία είναι ένα από τα συνηθέστερα μετρούμενα μεγέθη. Για το λόγο αυτό ο αριθμός των αισθητηρίων και των τρόπ ων μέτρησης είναι πολύ μεγάλος.

Οι γνωστότεροι παθητικοί αισθητήρες είναι τα θερμίστορς. Αυτά χωρίζονται σε δύο κατηγορίες ανάλογα με το θερμικό τους συντελεστή : Σε αρνητικού θερμικού συντελεστ ή (NTC) και σε θετικού θερμικού συντελεστή (PTC).

Τα θερμίστορ με αρνητικό θερμικό συντελεστή άγουν καλύτερα όσο αυξάνεται η θερμοκρασία, ενώ σε αυτά που έχουν θετικό συντελεστή συμβαίνει το αντίθετο. Τα θερμίστορ με αρνητικό συντελεστή θερμοκρασίας κατασκευάζονται από ημιαγώγιμο υλικό με προσμίξεις οξειδίων του σιδήρου. Τα θερμίστορ με θετικό συντελεστή κατασκευάζονται από ένα υλικό με βάση το τιτάνιο και κεραμικό υλικό. Η επίδραση των ιδιοτήτων του ημιαγωγικού υλικού με τις σιδηροηλεκτρικές ιδιότητες του τιτανίου δίνουν τα χαρακτηριστικά σε αυτά τα θερμίστορ. Υπ άρχουν ακόμα και θερμικοί αισθητήρες κατασκευασμένοι με πυρίτιο που έχουν το πλεονέκτημα του μικρού σφάλματος μέτρησης.

2.2.Πρακτικά Αισθητήρια Θερμοκρασίας.

Τα ICs αισθητήρια θερμοκρασίας, είναι εξαρτήματα αρκετά περίπλοκα, που μετατρέπουν την θερμοκ ρασία σε τάση. Η National Semiconductors είναι μια από τις κύριες εταιρίες που κατασκευάζουν αυτά τα ολοκληρωμένα και διαθέτει 4 αυτού του τύπου καθώς και παραλλαγές.

Το LM34CZ και LM35CZ είναι εξαρτήματα που οι έξοδοί τους είναι ανάλογες της θερμοκρασίας σε βαθμούς ^{o}F Και ^{o}C. Το LM3352 , είναι ένα πολύ απλό ρυθμιζόμενο ολοκληρωμένο, όπου η έξοδός του είναι ανάλογη της

θερμοκρασίας σε ^{o}C. Και τελικά το LM3911N που περιέχει μια τάση αναφοράς, ένα αισθητήρα θερμοκρασίας και ένα τελεστικό ενισχυτή. Το ολοκληρωμένο αυτό μπορεί να χρησιμοποιηθεί για την ένδειξη θερμοκρασίας ή σαν θερμοστάτης.

2.2.1. LM34CZ / LM35CZ

Τα LM34CZ και LM35CZ έχουν περίβλημα TO–92 με τρείς ακροδέκτες και καταναλώνουν ελάχιστο ρεύμα. Τα ολοκληρωμένα μπορούν να μας δώσουν έξοδο για την κλίμακα μετρήσεων που ξεκινάει –40^{o}C χρησιμοποιώντας δυο τροφοδοσίες.

Ένα μικρό μειονέκτημά τους είναι ο τείνουν στην αστάθεια όταν οι έξοδοί τους οδηγούν χωρητικά φορτία μεγαλύτερα από μερικά pF. Τα δυο ολοκληρωμένα μπορούν απ' ευθείας να χρησιμοποιηθούν σαν αναλογικά θερμόμετρα και αν συνδέσουμε τις εξόδους τους με όργανο των 100 mA. Επίσης μπορούν να οδηγήσουν ψηφιακό display με ευαισθησία 200mV αν συνδέσουμε την έξοδο ενός εξασθενητή 10:1.

2.2.2. PT100

Το θερμόμετρο PT100 αποτελείται σε γενικές γραμμές από ένα μετατροπέα αντίστασης σε τάση συνοδευόμενο από ψηφιακό βολτόμετρο 7106 με ένδειξη υγρών κρυστάλλων. Για μέτρηση θερμοκρασιών στην περιοχή 0…100 ^{o}C, ο μετατροπέας αντίστασης σε τάση απαιτεί μόνο μια απλή σταθεροποιημένη πηγή ρεύματος. Στην περιοχή αυτή η μεταβολή του συντελεστή (α) είναι ελάχιστη και συνεπώς η σχέση μεταξύ αντίστασης και θερμοκρασίας είναι πρακτικά γραμμική. Αντίθετα σε θερμοκρασίες μερικών εκατοντάδων βαθμών Κελσίου παρατηρούνται σημαντικές αποκλίσεις. Η ευθυγράμμιση των δυο μεγεθών θα μπορούσε βέβαια να γίνει πολύ εύκολα με κάποιον υπολογιστή, αλλά αυτό θα ανέβαζε πολύ το κόστος κατασκευής. Εκτός αυτού το θερμόμετρο θα ήταν πολύ δύσχρηστο.

Η πιο συμφέρουσα λύση από οικονομικής πλευράς είναι η διόρθωση του σφάλματος με ένα αναλογικό κύκλωμα. Η κλίση (ευαισθησία) του αισθητηρίου PT100 μειώνεται όσο αυξάνεται η θερμοκρασία.

Επιπλέον αν το θερμόμετρο Pt100 προορίζεται μόνο για χρήση σε μια συγκεκριμένη περιοχή θερμοκρασιών, μπορείτε να κάνετε ρύθμιση μόνο στην περιοχή που σας ενδιαφέρει. Έτσι θα επιτύχετε ακόμα μεγαλύτερη ακρίβεια. Ποτέ δεν πρέπει να ξεπερνάτε τα όρια θερμοκρασίας που δίνονται για το συγκεκριμένο αισθητήριο Pt100 που θα αγοράσετε. Αλλιώς, οι διαφορετικοί συντελεστές διαστολής του γυαλιού και του σύρματος πλατίνας μπορεί να οδηγήσουν το αισθητήριο στην καταστροφή.

ΚΕΦΑΛΑΙΟ 3

(Χαρακτηριστικά Αισθητήρων)

3.1. Η ακρίβεια

Η ακρίβεια είναι ίσως η βασικότερη στατική παράμετρος, γιατί καθορίζει το επί τοις εκατό σφάλμα. Στην πραγματικότητα η ακρίβεια καθορίζεται σαν ποσοστό σφάλματος.

Σφάλμα είναι η διαφορά μεταξύ της αληθούς (σωστής) τιμής της εξόδου από τον μετατροπέα και της πραγματικής του εξόδου που μετράμε εμείς. Η ακρίβεια λοιπόν εκφράζεται σαν ποσοστιαίο σφάλμα. Η ερώτηση που προκύπτει είναι : ποσοστό του τι;

Υπάρχουν τρεις διαφορετικές απαντήσεις. Η ακρίβεια μπορεί να εκφραστεί σαν ποσοστό της εξόδου πλήρους κλίμακας (% FSO), σαν ποσοστό της αναγνώσιμης τιμής, ή σε απόλυτες τιμές της εισόδου

3.2. Resolution (Διακριτικότητα ή ανάλυση)

Ο όρος 'Resolution' Διακριτικότητα ή ανάλυση, είναι η μικρότερη αλλαγή που μπορεί να διακρίνει η είσοδος ενός μετατροπέα ή ενός αισθητήρα και η οποία μπορεί να δώσει κάποια ανιχνεύσιμη αλλαγή στην έξοδό του. π.χ.+ 0.1m/sec ή 0,1 ^{0}C. Όσο μεγαλύτερη είναι η διακριτικότητα, τόσο μικρό τερο είναι το βήμα που μπορεί ο αισθητήρας να μετρά.

Για παράδειγμα ένας οπτικός κωδικοποιητής έχει ένα περιστρεφόμενο δίσκο με οκτώ τρύπες ανά 45^0 η κάθε μία. Ο κωδικοποιητής αυτός έχει και έναν οπτικό αισθητήρα που είναι προσαρμοσμένος πάνω στ ο δίσκο και μπορεί να περνά φως ανά 45^0 . Έτσι παράγεται ένας παλμός κάθε φορά που μία οπή ελευθερώνει τον οπτικό αισθητήρα και στη συνέχεια μετράται από ένα αθροιστή παλμών. Η μέτρηση δείχνει πόσο έχει περιστραφεί ο άξονας. Παλμός παράγεται μόνο κάθε 45^0. Ο αισθητήρας λοιπόν αυτός λέμε ότι έχει διακριτικότητα 45 ^{0}. Υπάρχουν βιομηχανικοί τύποι τέτοιων αισθητήρων για τη μέτρηση ταχύτητας αλλά και θέσης άξονα που διαθέτουν μέχρι και 2000 παλμούς ανά περιστροφή οπότε η διακριτικότητά τους είναι 360^0/2000 = 0,18^0 .

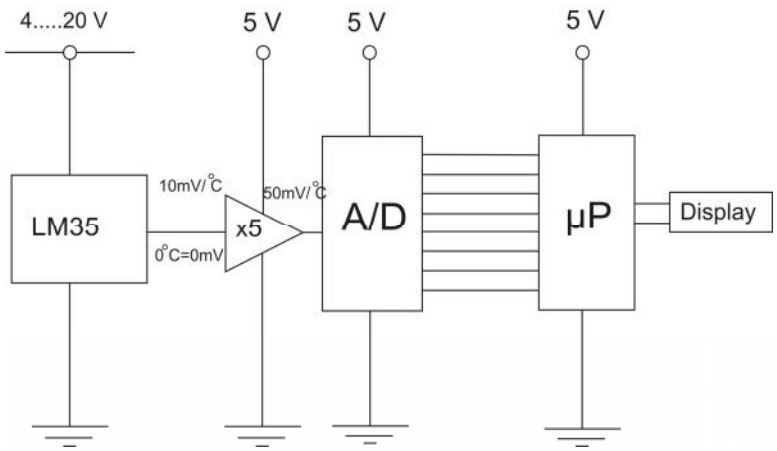
Οι υπολογιστές χρησιμοποιούνται στον βιομηχανικό έλεγχο σε συνδυασμό με αναλογικοψηφιακούς (A/D) μετατροπείς που χαρακτηρίζονται από ένα αριθμό Bits. Λέμε λοιπόν ότι κάποιος A/D μετατροπέας έχει 12 bit ανάλυση και εννοούμε ότι η αναλογική τάση που διαβάζει στην είσοδό του μεταφράζεται στην έξοδό του σε 2 ^{12} ή 4096 διαφορετικές ψηφιακές τιμές.

Παράδειγμα 1

Ένας αισθητήρας θερμοκρασίας LM35 δίνει 10 mV/ ^{0}C. Η έξοδός του λοιπόν θα είναι 240mV στους 24 ^{0}C. Θέλουμε να τον χρησιμοποιήσουμε για μέτρηση θερμοκρασίας στην περιοχή 0 - 100 ^{0}C με ένα μικροεπεξεργαστή.

(Ο κατασκευαστής μας δίνει και την τυπική τιμή της ακρίβειας του, 0,4 ^{0}C στους 25 ^{0}

Μπορούμε να χρησιμοποιήσουμε ένα A/D μετατροπέα των 8 bit με δυνατότητα 5 V στην πλήρη κλίμακα εισόδου για να διαβάσουμε τη θερμοκρασία και θέλουμε να έχουμε διακριτικότητα ανάγνωσης τουλάχιστον 1 ^{0}C .



Σχήμα 3.1

Η διακριτικότητα του μετατροπέα A/D (8 bit) είναι ένα μέρος στα 256 (ή 2 ^{8}).

Για 5 V πλήρη κλίμακα εισόδου κάθε bit αντιστοιχεί σε 5 V / 256 = **19,53 mV.**

Ο αισθητήρας θερμοκρασίας δίνει μόνο 10 mV /1 ^{0}C μεταβολή στην είσοδο του A/D και αυτός μπορεί να διακρίνει κατ' ελάχιστο 19,53mV.

Έτσι βλέπουμε ότι ο μετατροπέας A/D των 8 bit δεν δείνει καλή διακριτικότητα (~2 ^{0}C) για το πρόβλημά μας.

Επίσης στη μέγιστη θερμοκρασία των 100 ^{0}C θα έχουμε έξοδο από τον αισθητήρα 1V που είναι λίγο μπροστά στην δυνατότητα εισόδου του A/D (5V).

$$\frac{10mV}{^0C} X 100^0C = 1V$$

Η λύση λοιπόν είναι να ενισχύσουμε την έξοδο του αισθητήρα Χ 5 οπότε οι 100 ^{0}C θα αντιστοιχούν σε 5 V που αυτό ταιριάζει με την πλήρη κλίμακα εισόδου του μετατροπέα.

Τότε ένας βαθμός με ενίσχυση Χ 5 θα δίνει τάση εισόδου στον A/D :

Vo_{Amplif} = 1 ^{0}C x 10 mV/ ^{0}C x 5 = 50mV..

Τώρα ο μετατροπέας των 8 bit μας ικανοποιεί διότι η διακριτικότητά του θα είναι ~ 0,4 ^{0}C.

50 mV / 19,5 mV = 2,56 και 1/ 2,56 = 0,39 ^{0}C.

Εναλλακτικά μπορούμε να χρησιμοποιήσουμε και έναν 10 bit A/D οπότε η διακριτικότητά του θα είναι 1024 βήματα.

Κάθε bit τώρα του μετατροπέα, θα αντιστοιχεί σε 5 V/1024 = 4,8828 mV. Αυτό για την εφαρμογή μας αντιστοιχεί κατ' αρχή σε διακριτικό τητα ανάγνωσης του αισθητήρα μικρότερη από 0,5 ^{0}C.

Ωστόσο στους 100 ^{0}C ο αισθητήρας θα δώσει μόνο 488,28 mV.

100 X 4,8828 =488,28 V (~ 0,5 V)

Θα πρέπει λοιπόν να ενισχύσουμε Χ 10 για να εξαντλήσουμε την πλήρη κλίμακα (4,88V) εισόδου του A/D στους 100 ^{0}C..

Έτσι η έξοδος του αισθητήρα και με ενίσχυση Χ10 θα είναι :

Vo_{Amplif} = 1 ^{0}C x 10 mV/ ^{0}C x 10 = **100mV/ ^{0}C.**

Αλλά Vo_{Amplif} = 100 ^{0}C x 10 mV/ ^{0}C x 10 = **10V.**

Αφού όμως ο A/D δεν μπορεί να δεχτεί τα 10 V είσοδο, είμαστε αναγκασμένοι να χαμηλώσουμε την ενίσχυση για να μπορούμε να δούμε και τους 100 ^{0}C

Έτσι η έξοδος του αισθητήρα ενισχύεται Χ 5 και θα έχουμε :

Vo_{Amplif} = 1 ^{0}C x 10 mV/ ^{0}C x 5 = **50mV/^{0}C.**

Αλλά Vo_{Amplif} = 100 ^{0}C x 10 mV/ ^{0}C x 5 = **5V.**

Ο μετατροπέας λοιπόν των 10 bit μας δίνει πολύ καλύτερη διακριτικότητα ανάγνωσης, της τάξης των ~ 0,1 ^{0}C πόλυ καλύτερη από αυτήν που επιθυμούμε.

50 mV/ 4,88 mV = 10,24 mV

και 1/ 10,24 = 0,0976 ^{0}C.

Τετραπλασιάζοντας λοιπόν τα bit του A/D από 256 σε 1024 τετραπλασιάσαμε την διακριτικότητα ανάγνωσης του αισθητήρα από 0,4 σε 0.1 ^{0}C.

3.3. Υστέρηση

Είναι επίσης μια άλλη ένδειξη της αναπαραγωγικής ικανότητας της εξόδου του αισθητήρα. Μία συγκεκριμένη τιμή στην είσοδο, όταν φθάνει αυξανόμενα μπορεί να δίνει διαφορετική τιμή στην έξοδο, από όταν φθάνει ελαττούμενα από την πλήρη κλίμακα. Αυτό το έχουμε συναντήσει προηγούμενα στον αισθητήρα βάρους στο παράδειγμα 1. Επιβάλλεται λοιπόν η βαθμονόμηση να γίνεται και κατά τις δύο κατευθύνσεις, μία αυξάνοντας το ερέθισμα και μια ελαττώνοντας.

Αν δούμε τον πίνακα 1 στα 55 Kg , στη μία κατεύθυνση α ύξησης της φόρτισης έχουμε αποτέλεσμα στην έξοδο 9,85 mV. Μετά έχοντας φέρει τον μετατροπέα σε πλήρη κλίμακα ελαττώνοντας βρίσκουμε για το ίδιο βάρος, έξοδο 11,80 mV.

Για να υπολογίσουμε την υστέρηση παίρνουμε την απόλυτη διαφορά των δύο τιμών (11,8-9,85=0,95). Το ίδιο κάνουμε και σε κάθε σημείο του πίνακα. Η μεγαλύτερη από τις τιμές που καταγράφουμε δίνει την υστέρηση σαν ποσοστό της πλήρους κλίμακας εξόδου.

Η υστέρηση προκαλείται από ένα φυσικό μηχανισμό του μετατροπέα, παίρνοντας μια τιμή σαν αποτέλεσμα την ιστορίας του, δηλαδή των εισόδων που έχει λάβει. Η τιμή της υστέρησης που εμφανίζεται εξαρτάται από το μέγεθος των εισόδων που έχουν εφαρμοστεί στον μετατροπέα. Στη βαθμονόμηση πρέπει να τρέξουμε την είσοδο μέχρι την πλήρη κλίμακα πριν να επιστρ έψουμε στην ελάττωση των τιμών εισόδου. Στην πραγματικότητα αν η είσοδος δεν αυξηθεί μέχρι την πλήρη κλίμακα , η υστέρηση που υπολογίζουμε είναι μικρότερη από την κανονική.

Η αδράνεια και η τριβή προκαλούν ένα αποτέλεσμα που εμφανίζεται πολύ όμοιο με την υστέρηση. Αν δώσουμε τις τιμές σε μία είσοδο μετατροπέα με σάρωση

χωρίς να επιτρέπουμε στην έξοδο να σταθεροποιείται πριν από κάθε νέα μεταβολή της εισόδου, η αδράνεια και η τριβή μπορούν να κάνουν της έξοδο να καθυστερήσει. Έτσι η καμπύλη μεταφοράς του μετατροπέα (είσοδος ως προς έξοδο) φαίνεται σαν γράφημα υστέρησης. Το αποτέλεσμα είναι χειρότερο όσο αυξάνεται η ταχύτητα σάρωσης της εισόδου. Αυτό όμως δεν είναι υστέρηση.

3.4. Ορολογία Για Αισθητήρες

Κάποιοι από τους ορισμούς μπορεί να έχουν διαφορετικές σημασίες σε διαφορετικά πεδία ή και να μην εφαρμόζονται όλοι για κάποιο είδος αισθητήρα/ μετατροπέα.

Συνοψίσουμε εδώ τούς τεχνικούς όρους και τις έννοιες που χρησιμοποιούνται συχνά για να περιγράψουν τα αισθητήρια, τους μετατροπείς μεγεθών, τα όργανα μέτρησης, και την απόκρισή τους.

- **Αισθητήρας μέτρησης (sensor):** είναι ένα ειδικό στοιχείο αίσθησης που αντιδρά σε αλλαγές του περιβάλλοντος (π.χ. αισθητήριο πίεσης, θερμοκρασίας, υγρασίας κλπ)

- **Μετατροπέας ενέργειας (Transducer):** είναι συσκευή για την μετατροπή ενέργειας από μια μορφή σε άλλη (π.χ. μετατροπή περιστροφικής κίνησης ανεμομέτρου σε ηλεκτρικούς παλμούς, μετατροπή θερμοκρασίας σε τάση κλπ). Σήμερα πολλοί μετατροπείς είναι ενσωματωμένοι μέσα στους αισθητήρες και η έννοια έχει σχεδόν ταυτοποιηθεί με τον αισθητήρα .

- **Συλλέκτης δεδομένων (Data Logger or data processor):** είναι συσκευή που δέχεται εξόδους από διάφορα αισθητήρια ή Transducers και τα μετατρέπει σε μορφή που να μπορούν να αποθηκευτούν.

- **Σύστημα απόκτησης δεδομένων (Data Acquisition System):** είναι πλήρες σύστημα με αισθητήρια, transducers, Data Logger, μνήμη , κατάλληλο Software και πιθανόν πομπό επικοινωνίας.

- **Βαθμονόμηση (calibration):** Η έννοια αναφέρεται στις μονάδες που βαθμολογείται η κλίμακα εμφάνισης ή καταγραφής ενός οργάνου. Για παράδειγμα αν ένας αισθητήρας που μετρά ταχύτητα, παράγει μια ηλεκτρική έξοδο σε μορφή τάσης , ο δείκτης του οργάνου κινείται ανάλογα με την τάση που εφαρμόζεται σ΄ αυτόν , αλλά η θέση του χαρακτηρίζεται από κάποια τιμή ταχύτητας και όχι τάσης . Λέμε λοιπόν ότι το ταχύμετρο βαθμονομείται ως προς την ταχύτητα.

- **Ανεμογράφος ή ανεμόμετρο (Wind measuring system):** είναι σύστημα που περιλαμβάνει αισθητήρια και Transducers, Data Logger, πρόγραμμα συλλογής δεδομένων και ιστό στήριξης, κατάλληλο για τη μέτρηση ταχύτητας και διεύθυνσης του ανέμου.

- **Ανεμόμετρο (velocity sensor):** είναι αισθητήρας που μετατρέπει την περιστροφική κίνηση σε τάση ανάλογη της ταχύτητας του ανέμου ή και σε ένα αριθμό παλμών /sec.

- **Ανεμοδείκτης (Direction Sensor):** είναι αισθητήρας που μετατρέπει την κίνηση ενός ανεμουρίου σε τάση ανάλογη της διεύθυνσης του ανέμου.

- **Σφάλμα (error):** είναι η διαφορά ανάμεσα στην μετρούμενη τιμή μιας ποσότητας και την πραγματική τιμή. Συχνά εκφράζεται και επί τοις εκατό οπότε αντιπροσωπεύει την ακρίβεια του συστήματος.

- **Σφάλμα μέτρησης (Error):** είναι η αλγεβρική διαφορά μεταξύ της τιμής ένδειξης και της πραγματικής τιμής του μετρούμενου μεγέθους , π.χ. + 0.1 m/s.

- **Ακρίβεια οργάνου (Accurancy):** είναι ο βαθμός με τον οποίο ένα όργανο μπορεί να μετρήσει μια μεταβλητή και σε σύγκριση με το πραγματικό της μέγεθος , π.χ. +/- 0.2 m/s στα 10 m/s. Η ακρίβεια δηλαδή εκφράζει το επί τοις εκατό σφάλμα ως προς το εύρος μέτρησης της συσκευής ή του αισθητήρα.

- **Διακριτικότητα ή ανάλυση (Resolution):** είναι η μικρότερη μεταβολή στο περιβάλλον(ή στην είσοδο ενός μετατροπέα), που προκαλεί μια ανιχνεύσιμη μεταβολή στην ένδειξη ενός αισθητήρα ή οργάνου π.χ.+ 0.1m/sec. Όσο μεγαλύτερη είναι, τόσο μικρότερο είναι το βήμα που μπορεί ο αισθητήρας να μετρά.

- **Ευαισθησία (Sensitivity):** είναι ο λόγος των τιμών πλήρους κλίμακας εξόδου προς εισόδου, κάτω από συγκεκριμένες συνθήκες και είναι διάφορο της Διακριτικότητας, π.χ. 10 παλμοί ανά m/s ή Volt/mm.

- **Ευστάθεια (stability):** είναι το μέτρο της μεταβολής της εξόδου μιας συσκευής, όταν η είσοδος και οι συνθήκες παραμένουν σταθερά , κατά τη διάρκεια μιας μεγάλης χρονικής περιόδου.

- **Εύρος (operating range) :** είναι τα όρια που μπορεί η συσκευή να λειτουργεί αξιόπιστα. Εκφράζεται με την μέγιστη και την ελάχιστη τιμή που μπορεί να μετρά.

- **Αξιοπιστία οργάνου (Reliability):** είναι μια μέτρηση της πιθανότητας λειτουργίας του μέσα σε περιορισμένα όρια σφάλματος, για δεδομένη χρονική περίοδο και κάτω από συγκεκριμένες συνθήκες. Είναι συγγενής έννοια με το χρόνο λειτουργίας.

- **Χρόνος ζωής ή λειτουργίας (operating life):** είναι ο ωφέλιμος χρόνος λειτουργίας ενός αισθητήρα για να λειτουργεί στα πλαίσια των προδιαγραφών του. Εκφράζεται σε μονάδες χρόνου ή με τον αριθμό των κύκλων λειτουργίας που μπορεί να διεκπεραιώσει επιτυχώς (π.χ. ένα ρελέ έχει 108 επαναλήψεις με το ονομαστικό του φορτίο).

- **Γραμμικότητα (linearity):** αποτελεί το βαθμό στον οποίο η γραφική παράσταση της εξόδου ως προς την είσοδο του αισθητήρα προσεγγίζει μια ευθεία γραμμή. Η γραμμικότητα ενός αισθητήρα μπορεί να υπάρχει μόνο για μία περιοχή τιμών εξόδου(π.χ. από 10-125 0C) Μπορεί να εκφράζεται ως προς το μέγιστο βαθμό απόκλισης από την ευθεία γραμμή σε όλο το εύρος των τιμών εισόδου (ποσοστό επί του εύρους λειτουργίας ή της πλήρους κλίμακάς του).

- **Επαναληψιμότητα οργάνου (Repeatability or Reproducibility):** είναι η πλησιέστερη συμφωνία, μεταξύ ενός αριθμού διαδοχικών μετρήσεων του ίδιου μεγέθους, κάτω από τις ίδιες συνθήκες. Εκφράζεται ως ένα +/ - % μέγιστο ποσοστό επί της ένδειξης, ή ως τα όρια ακρίβειας +/- κάθε ένδειξης. Κατά άλλους είναι το πόσο καλά η έξοδος ενός μετατροπέα επιστρέφει σε μια δεδομένη τιμή, όταν η ίδια είσοδος εφαρμοστεί μερικές φορές. Προσοχή δεν την συγχέουμε με την ακρίβεια .

- **Υστέρηση (Hysteresis) :** προκαλεί διαφορές στην έξοδο που δίνει ένας αισθητήρας , όταν η κατεύθυνση μεταβολής της εισόδου αντιστραφεί. Με της υστέρηση παράγεται σφάλμα και επηρεάζεται η ακρίβεια της συσκευής. Σε μηχανικά συστήματα προκαλείται από παράγοντες όπως η μηχανική τάση και η τριβή.

- **Καθυστέρηση (Lag):** είναι η καθυστέρηση της αλλαγής της τιμής εξόδου ενός αισθητήρα ως προς την αλλαγή της εισόδου του. Στον έλεγχο μπορεί να επηρεάσει σημαντικά την απόδοση.

- **Απόκριση (response):** είναι ο χρόνος που απαιτεί μια συσκευή για να λάβει την τελική τιμή εξόδου της για μια δεδομένη είσοδο. Πολλές φορές εκφράζεται και σαν ποσοστό επί της τελικής τιμής εξόδου. Π.χ. λέμε ότι η συσκευή χρειάζεται 3 sec για να λάβει η έξοδός της το 95% της τελικής της τιμής.

- **Ταχύτητα απόκρισης (Speed of Response):** είναι ο χρόνος που μεσολαβεί μετά από μια απότομη μεταβολή στην είσοδο ενός συστήματος, μέχρι η έξοδος να σταθεροποιηθεί ή να φθάσει το 90 % της πλήρους μεταβολής της.

- **Σταθερά Χρόνου (Time Constant):** είναι η χρονική περίοδος που απαιτείται για ένα αισθητήρα, ώστε να ανταποκριθεί στο 63.2 % (1-1/e) της τελικής του τιμής. Στις μετεωρολογικές μετρήσεις για παράδειγμα η ένδειξη προκαλείται από άλλους περιβαλλοντικούς παράγοντες και αυτοί πρέπει να καθορίζονται.

- **Σταθερά Απόστασης (Distance Constant) :** είναι το μήκος ρευστού που πρέπει να περάσει από ένα αισθητήριο για να αποκριθεί αυτό στο 63.2 % της μεταβολής της ταχύτητας του ρευστού.

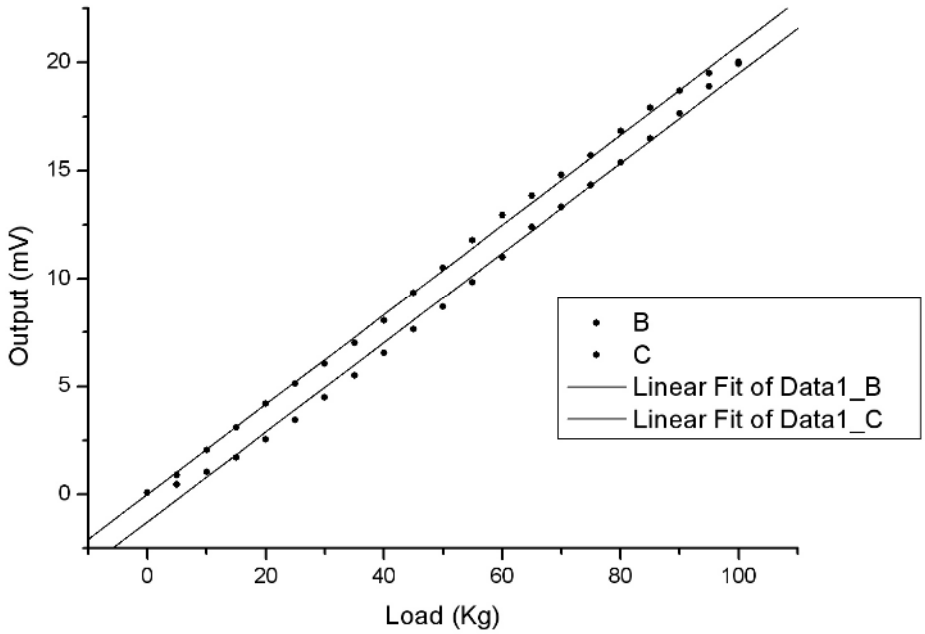
- **Λόγος εξομάλυνσης (Damping Ratio):** χρησιμοποιείται για παράδειγμα στους αισθητήρες διεύθυνσης ανέμου που έχουν δευτέρας τάξης απόκριση στο σήμα διέγερσης.

- **Νεκρή Ζώνη (Dead Band or zone) :** είναι η περιοχή μέσα στην οποία μπορεί να μεταβάλλεται η είσοδος χωρίς να προκαλεί μεταβολή στο σήμα εξόδου του οργάνου, π.χ. στα συνηθισμένα αισθητήρια διεύθυνσης ανέμου με κυκλικό ποτενσιόμετρο έχουμε 3 μοίρες Dead Band. Η νεκρή ζώνη προκύπτει συνήθως λόγω

στατικής τριβής ή και υστέρησης σε άλλα αισθητήρια ή συστήματα και δεν είναι απαραίτητο να υπάρχει πάντα σε όλο το εύρος ενός οργάνου.

- **Ολίσθηση (Drift):** είναι η φυσική τάση μιας συσκευής ή ενός συστήματος να μεταβάλλει τα χαρακτηριστικά του με το χρόνο, ή λόγω περιβαλλοντικών μεταβολών ή ακόμα και για διάφορους άλλους λόγους. Η ολίσθηση βέβαια επηρεάζει την ακρίβεια. Πολύ συνηθισμένο αίτιο είναι η αλλαγή της θερμοκρασίας του περιβάλλοντος που επηρεάζει όλα τα ηλεκτρονικά εξαρτήματα και την τιμή τους, όπως επίσης και η γήρανση, η οξείδωση για άλλα, ή και η διάβρωση .

- **Ηλεκτρικός Θόρυβος (noise):** ονομάζεται η παρουσία ανεπιθύμητων ηλεκτρικών σημάτων που μπορούν να αλλοιώσουν ή και να αποκρύψουν το σήμα που μεταφέρει τη χρήσιμη πληροφορία.



Σχήμα 3.2

Κεφάλαιο 4

(Λήψη και Αποθήκευση Μετρήσεων σε USB Flash)

4.1. Γενική Περιγραφή της συσκευής

Σκοπός της κατασκευής μου είναι η λήψη μετρήσεων και η αποθήκευσή τους σε USB Flash για μεγάλο χρονικό διάστημα.

Ο μικροεπεξεργαστής που χρησιμοποιείται είναι ο Atmel mega128,ο οποίος διαθέτει έναν 10bit μετατροπέα αναλογικού σήματος σε ψηφιακό .Σε κάθε ένα από τα 8 Κανάλια του ADC μπορεί να συνδεθεί ένα αισθητήριο θερμοκρασίας , το οποίο θα παρέχει μία αναλογική τάση στον ADC του μικροεπεξεργαστή. Ο ADC μετράει αυτήν την τάση ανά 5 sec σε κάθε κανάλι και την μετατρέπει σε μία ψηφιακή τιμή. Στην συνέχεια ο μικροεπεξεργαστής λαμβάνει την ψηφιακή τιμή και την αποθηκεύει στο USB Flash.

Στα 4 πρώτα Κανάλια της συσκευής έχουν συνδεθεί 4 αισθητήρια θερμοκρασίας ενώ τα υπόλοιπα 4 είναι ελεύθερα. Τα αισθητήρια θερμοκρασίας που χρησιμοποιούνται είναι τα AD22100 της Analog Devices.

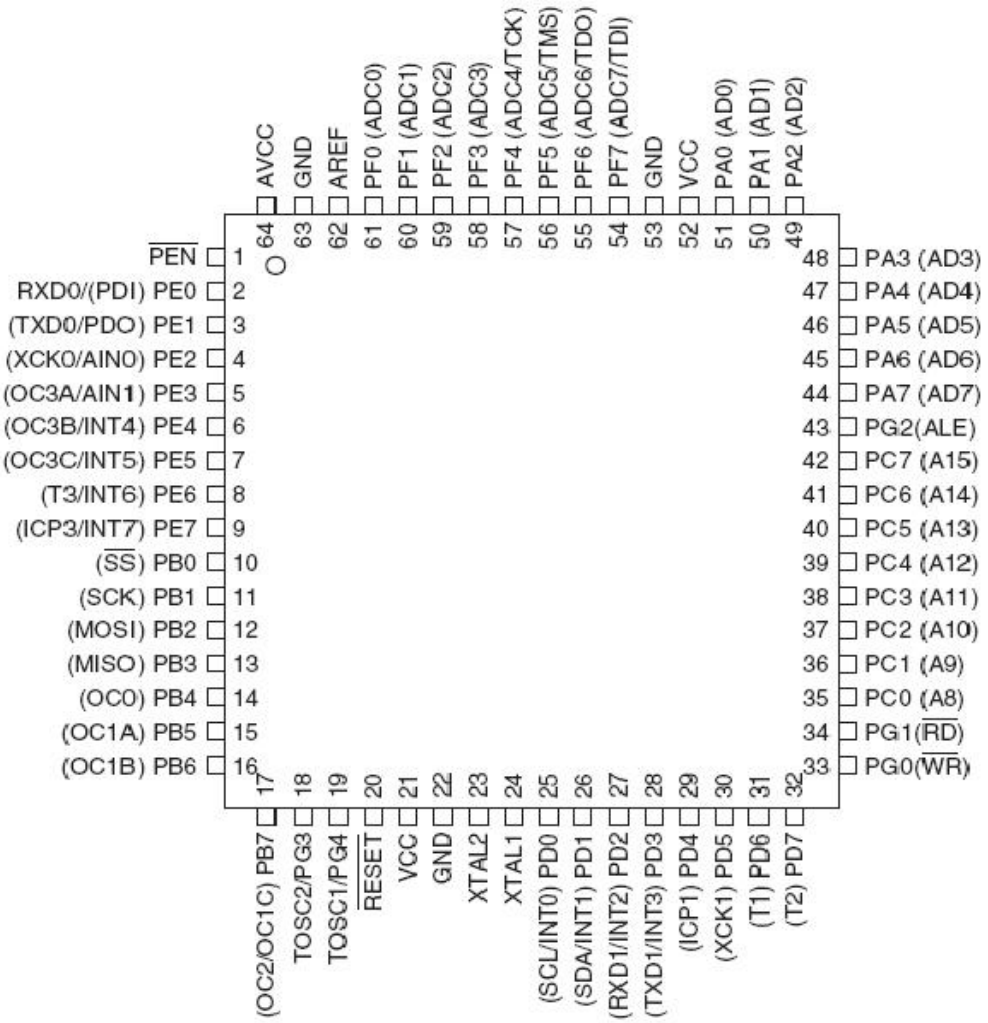
Για την απεικόνιση των λειτουργιών της συσκευής έχει χρησιμοποιηθεί μια οθόνη υγρών κρυστάλλων 4X20 (Liquid Crystal Display), ενώ για την επιλογή των λειτουργιών και για το αρχικό Setup χρησιμοποιούνται 16 πλήκτρα. Όταν η συσκευή είναι σε λειτουργία, παρουσιάζονται στο Display οι τιμές των θερμοκρασιών στα 4 κανάλια και ταυτόχρονα αποθηκεύονται στην εξωτερική Flash card. Κάποια στιγμή όταν τελειώσουμε τις μετρήσεις μπορούμε να μεταφέρουμε τις τιμές από την Flash ένα PC.

Για τον προγραμματισμό του μικροεπεξεργαστή υπάρχει πάνω στην πλακέτα ένα βύσμα από το οποίο γίνεται ο σειριακός προγραμματισμός χωρίς να χρειάζεται η απομάκρυνση του μικροεπεξεργαστή από την πλακέτα της κατασκευής (ISP-In System Programming).

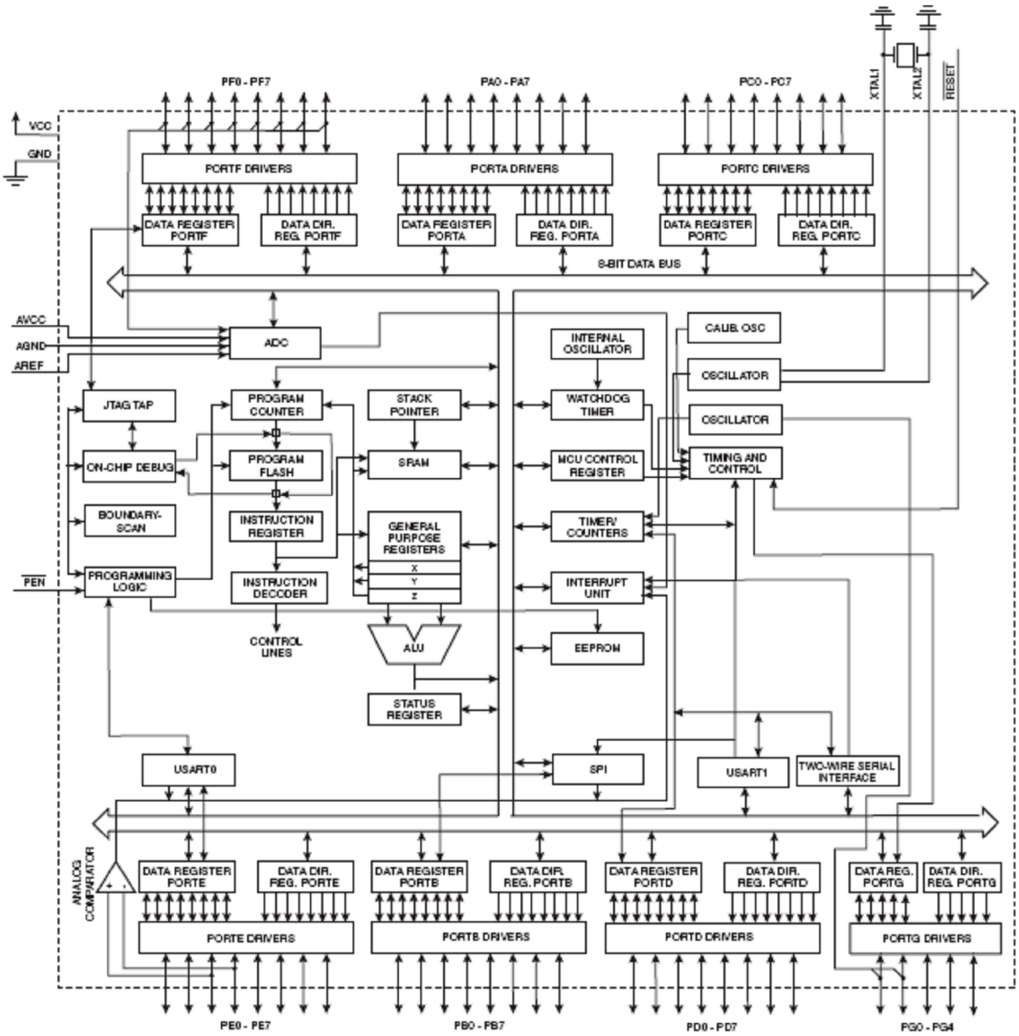
Η πλακέτα τροφοδοτείται από ένα τροφοδοτικό με τάση 9 Volts η οποία σταθεροποιείται στα 5 Volts για την τροφοδοσία των κυκλωμάτων της μέσω του LM7805,το οποίο αρκεί για φορτία μέχρι 800mA και απαιτεί ελάχιστη τάση εισόδου 7,5 Volts.

4.2. Ο ATMEL mega 128

Ο Atmel mega 128 είναι ένας 8-bit μικροϋπολογιστής αρχιτεκτονικής RISC, διαθέτει 53 I/O προγραμματιζόμενα pin, η τροφοδοσία του είναι στα 5V και χρησιμοποιεί κρύσταλλο στα 16 MHz. Διαθέτει μνήμη προγράμματος 128Kbytes, μνήμη EEPROM 4KBytes και εσωτερική SRAM 4KBytes.



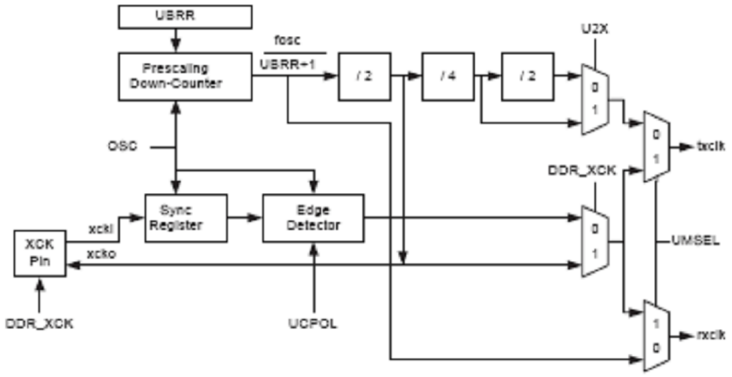
Εικόνα 4.1 AT mega128



Εικόνα 4..2. Το Blog Diagram του Atmel mega 128

4.3. USART (Σειριακός Προγραμματισμός)

Ο Atmel mega 128 διαθέτει δύο σειριακές θύρες επικοινωνίας οι οποίες είναι η USART0 (PE0, PE1) και USART1 (PD2, PD3). Η USART μπορεί να λειτουργήσει είτε σύγχρονα, είτε ασύγχρονα. Στην κατασκευή έχει επιλεχθεί ο ασύγχρονος τρόπος επικοινωνίας με 8 bit δεδομένων, 2 stop bit και κανένα bit ισοτιμίας (parity bit). Η επιλεγμένη ταχύτητα μεταφοράς των δεδομένων είναι 9600 bps (bit ανά δευτερόλεπτο).



Εικόνα 4.3 Clock Generation Logic,Block Diagram

4.4. USART Register Description

4.4.1. USARTn I/O Data Register-UDRn

Bit	7	6	5	4	3	2	1	0	
	RXBn[7:0]								UDRn (Read)
	TXBn[7:0]								UDRn (Write)
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Ο USARTn Transmit Data Register και ο USARTn Receive Data Register μοιράζονται την ίδια I/O διεύθυνση, καλούμενη ως USARTn Data Register ή ως UDRn. Ο Transmit Data Buffer Register (TXBn) είναι ο προορισμός για τα δεδομένα που γράφονται στην UDRn θέση register. Η ανάγνωση της θέσεως των register UDRn, θα επιστρέψει το περιεχόμενο του Receive Data Buffer Register (RXBn). Για τους 5, 6, ή τους 7 bits χαρακτήρες, τα ανώτερα αχρησιμοποίητα bits θα αγνοηθούν από τον Transmitter και θα τεθούν στο 0 από τον Receiver.

Ο Transmit buffer μπορεί να γραφεί μόνον όταν η σημαία UDREn στον register UCSRAn έχει τεθεί. Τα δεδομένα που γράφονται στον UDRn όταν η σημαία UDREn δεν έχει τεθεί στον 1, θα αγνοηθούν από τον USARTn Transmitter. Όταν τα δεδομένα γραφούν στον Transmitter Buffer και ο Transmitter είναι ενεργοποιημένος, ο Transmitter θα φορτώσει τα δεδομένα

στον Transmit Shift Register όταν ο Shift Register είναι άδειος. Τότε τα δεδομένα θα αποσταλούν σειριακά στον ακροδέκτη TXDn.

Ο Receive Buffer αποτελείται από μία μνήμη FIFO δύο επιπέδων. Η μνήμη FIFO θα αλλάξει την κατάσταση της όποτε ο Receive Buffer διαβάζεται.

4.4.2. USART Control And Status Register A-UCSRnA

Bit	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

i) ***Bit 7 – RXCn: USART Receive Complete***

Αυτό το bit τίθεται στο 1 όταν υπάρχουν αδιάβαστα δεδομένα στον Receive Buffer και τίθεται στο 0 όταν ο Receive Buffer είναι άδειος δηλαδή έχουν διαβαστεί τα δεδομένα του. Εάν ο receiver είναι εκτός λειτουργίας, το bit RXCn θα είναι μηδέν. Η σημαία του RXCn μπορεί να χρησιμοποιηθεί για να παράγει Receive Complete Interrupt δηλαδή ένα interrupt που υποδηλώνει ότι η λήψη των δεδομένων έχει ολοκληρωθεί.

ii) ***Bit 6 – TXCn: USART Transmit Complete***

Αυτό το bit τίθεται στο 0 όταν όλο το περιεχόμενο του Transmitt Shift Register έχει διαβαστεί και δεν έχει εμφανιστεί κανένα νέο δεδομένο κατά την ίδια την περίοδο στον Transmit Buffer (UDRn). Η σημαία TXCn bit τίθεται στο 0 αυτόματα όταν ένα Transmit Complete Interrupt έχει εκτελεστεί δηλαδή ένα interrupt που υποδηλώνει ότι η αποστολή των δεδομένων έχει ολοκλη ρωθεί ή μπορεί να καθαριστεί με το γράψιμο ενός 1 στην θέση αυτού του bit.

iii) ***Bit 5 – UDREn: USART Data Register Empty***

Η σημαία UDREn δείχνει εάν ο Transmit Buffer (UDRn) είναι έτοιμος να λάβει νέα δεδομένα. Εάν το UDREn είναι 1, ο buffer είναι άδειος και επ ομένως έτοιμος να γραφτεί. Η σημαία UDREn μπορεί να παράγει ένα Data Register Empty Interrupt δηλαδή ένα interrupt που υποδηλώνει ότι ο data register είναι άδειος. Το UDREn τίθεται στο 1 μετά από ένα reset για να δείξει ότι ο Transmitter είναι έτοιμος.

iv) ***Bit 4 – FEn: Frame Error***

Αυτό το bit τίθεται στο 1 εάν ο επόμενος χαρακτήρας στον Receive Buffer είχε κάποιο σφάλμα κατά την λήψη του. Για παράδειγμα όταν το πρώτο stop bit του επόμενου χαρακτήρα στον Receive Buffer είναι μηδέν. Αυτό το bit είναι έγκυρο έως ότου ο Receive Buffer (UDRn) διαβαστεί. Το bit FEn είναι μηδέν όταν stop bit των λαμβανόμενων δεδομένων είναι ένα. Αυτό το bit πρέπει πάντα να τίθεται σε μηδέν, όταν γράφεται ο UCSRnA.

v) ***Bit 3 – DORn: Data OverRun***

Αυτό το bit τίθεται στο 1 όταν μία Data OverRun κατάσταση ανιχνευτεί. Μια data overrun εμφανίζεται όταν ο Receive Buffer είναι πλήρης (δύο χαρακτήρες), ένας νέος χαρακτήρας περιμένει στον Receive Shift Register και ένα νέο start bit ληφθεί. Αυτό το bit είναι έγκυρο έως ότου ο Receive Buffer (UDRn) διαβαστεί. Αυτό το bit πρέπει πάντα να τίθεται σε μηδέν, όταν γράφεται ο UCSRnA.

vi) ***Bit 2 – UPEn: Parity Error***

Αυτό το bit τίθεται στο 1 εάν ο επόμενος χαρακτήρας στον Receive Buffer είχε κάποιο Parity Error (σφάλμα ισοτιμίας) όταν λήφθηκε και το Parity Checking (έλεγχος ισοτιμίας) ενεργοποιήθηκε σε αυτό το σημείο (UPMn1=1).Αυτό το bit είναι έγκυρο έως ότου ο Receive Buffer (UDRn) αναγνωστεί. Αυτό το bit πρέπει πάντα να τίθεται σε μηδέν.

vii) ***Bit 1 – U2Xn: Double the USART Transmission Speed***

Αυτό το bit έχει επίδραση μόνον στην ασύγχρονη λειτουργία. Στην σύγχρονη λειτουργία θα πρέπει να γραφεί στο με 0. Το γράψιμο αυτού του bit με 1 θα μειώσει τον διαιρέτη του baudrate από 16 σε 8 διπλασιάζοντας αποτελεσματικά το ρυθμό μεταφοράς για την ασύγχρονη επικοινωνία.

viii) ***Bit 0 – MPCMn: Multi-Processor Communication Mode***

Αυτό το bit ενεργοποιεί τον Multi-processor Communication mode. Όταν το bit MPCMn γραφεί με 1, όλα τα εισερχόμενα frames που παραλαμβάνονται από τον USART Receiver που δεν περιέχουν πληροφορίες διευθύνσ εων θα αγνοηθούν.

4.4.3. USARTn Control and Status Register B – UCSRnB

Bit	7	6	5	4	3	2	1	0	
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n	UCSRnB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

i) ***Bit 7 – RXCIEn: RX Complete Interrupt Enable***

Το γράψιμο αυτού του bit στο 1 ενεργοποιεί το interrupt στην σημαία RXC. Ένα USART Receive Complete interrupt θα παραχθεί μόνον εάν το bi t RXCIE έχει γραφεί με 1, η σφαιρική σημαία interrupt στον SREG έχει γραφεί με 1 και το bit RXC στον UCSRnA έχει τεθεί στο 1.

ii) ***Bit 6 – TXCIE: TX Complete Interrupt Enable***

Το γράψιμο αυτού του bit με 1 ενεργοποιεί το interrupt στη σημαία TXCn. Ένα USARTn Transmit Complete Interrupt θα παραχθεί μόνον εάν το bit TXCIEn έχει γραφεί με 1, η σφαιρική σημαία interrupt στον SREG έχει γραφεί με 1 ’ και το bit TXCn στον UCSRnA έχει τεθεί στο 1.

iii) ***Bit 5 – UDRIEn: USART Data Register Empty Interrupt Enable***

Το γράψιμο αυτού του bit με 1 ενεργοποιεί το interrupt στη σημαία UDREn. Ένα Data Register Empty Interrupt θα παραχθεί μόνον εάν το bit UDRIEn έχει γραφεί με 1, η σφαιρική σημαία interrupt στον SREG έχει γραφεί 1 και το bit UDREn στον UCSRnA έχει τεθεί στο 1.

iv) ***Bit 4 – RXENn: Receiver Enable***

Το γράψιμο αυτού του bit με 1 ενεργοποιεί τον USARTn Receiver. Όταν ο Receiver ενεργοποιηθεί θα σταματήσει την κανονική λειτουργία του ακροδέκτη RxDn και θα τον χρησιμοποιήσει για την χρήση της USART. Απενεργοποιώντας τον Rece iver οι σημαίες FEn, DORn και UPEn ακυρώνονται.

v) ***Bit 3 – TXENn: Transmitter Enable***

Το γράψιμο αυτού του bit με 1 ενεργοποιεί τον USARTn Transmitter. Όταν ο Transmitter ενεργοποιηθεί θα σταματήσει την κανονική λειτουργία του ακροδέκτη TxDn και θα τον χρησιμοποιήσει για την χρήση της USART. Η απενεργοποίηση του Transmitter (γράφοντας στο TXENn μηδέν) δεν θα γίνει αποτελεσματική μέχρι να ολοκληρωθούν οι τρέχουσες μεταδόσεις που εκκρεμούν, δηλαδή όταν ο Transmit Shift Register και ο Transmit Buffer Register δεν περιέχουν δεδομένα προς μεταφορά. Όταν ο Transmitter απενεργοποιείται ο ακροδέκτης TxDn παύει να χρησιμοποιείται από την Transmit λειτουργία και επιστρέφει στην κανονική του λειτουργία.

vi) ***Bit 2 – UCSZn2: Character Size***

Τα bits UCSZn2 που συνδυάζονται με το bit UCSZn1:0 στον UCSRnC register θέτουν τον αριθμό δεδομένων (μέγεθος χαρακτήρα) ενός frame που Receiver και ο Transmitter χρησιμοποιούν.

vii) ***Bit 1 – RXB8n: Receive Data Bit 8***

Το RXB8n είναι το ένατο bit των δεδομένων του λαμβανόμενου χαρακτήρα κατά λειτουργία των 9-data bits. Θα πρέπει να διαβαστεί πριν διαβαστούν τα low bits από τον UDRn.

viii) ***Bit 0 – TXB8n: Transmit Data Bit 8***

Το TXB8n είναι το ένατο bit των δεδομένων του προς αποστολή χαρακτήρα κατά λειτουργία των 9-data bits. Θα πρέπει να εγγραφεί πριν εγγραφούν τα low bits από τον UDRn.

4.4.4. USARTn Control and Status Register C – UCSRnC

Bit	7	6	5	4	3	2	1	0	
	–	UMSELn	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

i) ***Bit 7 – Reserved Bit***

Αυτό το bit διατηρείται για μελλοντική χρήση. Για την συμβατότητα με μελλοντικές συσκευές, αυτό το bit πρέπει να γράφεται στο μηδέν όταν ο UCSRnC εγγράφεται.

ii) ***Bit 6 – UMSELn: USART Mode Select***

Αυτό το bit επιλέγει μεταξύ του ασύγχρονου και σύγχρονου τρόπου λειτουργίας.

UMSELn	Mode
0	Asynchronous Operation
1	Synchronous Operation

Πίνακας 4.4 *UMSELn bit Settings*

iii) ***Bit 5:4 – UPMn1:0: Parity Mode***

Αυτά τα bits ενεργοποιούν και θέτουν τον τύπο παραγω γής και ελέγχου της ισοτιμίας. Εάν ενεργοποιηθεί, ο Transmitter θα παραγάγει αυτόματα και θα στείλει την ισοτιμία των σταλμένων δεδομένων μέσα σε κάθε frame. Ο Receiver θα παραγάγει μία τιμή ισοτιμία για τα εισερχόμενα δεδομένα και θα την συγκρίνει με την ρύθμιση του UPMn0. Εάν ένας κακός συνδυασμός ανιχνευτεί, η σημαία UPEn στον UCSRnA θα τεθεί στο 1.

UPMn1	UPMn0	Parity Mode
0	0	Disabled
0	1	(Reserved)
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

Πίνακας 4.5 *UPMn bits Settings*

iv) ***Bit 3 – USBSn: Stop Bit Select***

Αυτό το bit επιλέγει τον αριθμό των Stop Bits που παρεμβάλλονται από τον Transmitter. Ο Receiver αγνοεί αυτήν την ρύθμιση.

USBSn	Stop Bit(s)
0	1-bit
1	2-bits

Πίνακας 4.6 USBSn bit Settings

v) ***Bit 2:1 – UCSZn1:0: Character Size***

Τα bits UCSZn1:0 που συνδυάζονται με το bit UCSZn2 στον UCSRnB register, θέτουν τον αριθμό των bits των δεδομένων (μέγεθος χαρακτήρα) σε ένα frame για την χρήση από τον Receiver και τον Transmitter.

UCSZn2	UCSZn1	UCSZn0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

Πίνακας 4.7 *UCSZn bits Settings*

vi) ***Bit 0 – UCPOLn: Clock Polarity***

Αυτό το bit χρησιμοποιείται για στον σύγχρονο τρόπο λειτουργίας μόνο.

4.4.5. *USART Baud Rate Registers - UBRRnL and UBRRnH*

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	UBRRn[11:8]				UBRRnH
	UBRRn[7:0]								UBRRnL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

i) ***Bit 15:12 – Reserved Bits***

Αυτά τα bits κρατούνται για μελλοντική χρήση. Για την συμβατότητα με μελλοντικές συσκευές, αυτά τα bits πρέπει να γραφούν στο μηδέν όταν γράφεται ο UBRRnH.

ii) **Bit 11:0 – UBRRn11:0: USARTn Baud Rate Register**

Αυτός είναι ένας 12-bit register που περιέχει το baudrate της εκάστοτε USARTn. Ο UBRRnH περιέχει τα τέσσερα σημαντικότερα bits και ο UBRRnL περιέχει τα οκτώ λιγότερο σημαντικά bits του USARTn baudrate. Μεταδόσεις δεδομένων από τον Receiver και τον Transmitter θα διακοπούν εάν το baudrate αλλάξει. Το γράψιμο του UBRRnL θα προκαλέσει μία άμεση αναπροσαρμογή του baudrate.

4.5. ADC (Analog to Digital Converter)

The ATmega128 features a 10-bit successive approximation ADC. The ADC is connected to an 8-channel Analog Multiplexer which allows 8 single-ended voltage inputs constructed from the pins of Port F. The ADC has a separate analog supply voltage pin, AVCC. AVCC must not differ more than ± 0.3V from V_{CC}.Internal reference voltages of nominally 2.56V or AVCC are provided On-chip. The volt- age reference may be externally decoupled at the AREF pin by a capacitor for better noise performance.

After the conversion is complete (ADIF is high), the conversion result can be found in the ADC Result Registers (ADCL, ADCH). For single ended conversion, the result is :

$$ADC = \frac{V_{IN} \cdot 1024}{V_{REF}}$$

where V_{in} is the voltage on the selected input pin and V_{ref} the selected voltage reference 3.3 Volts in this case. .

4.6. ADC Register Description

4.6.1. ADC Multiplexer Selection Register – ADMUX

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

i) Bit 7:6 – REFS1:0: Reference Selection Bits

These bits select the voltage reference for the ADC, as shown in Table 97. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set). The internal voltage reference options may not be used if an external reference voltage is being applied to the AREF pin.

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal Vref turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

Πίνακας 4.8 *Voltage Reference Selections for ADC*

ii) Bit 5 – ADLAR: ADC Left Adjust Result

The ADLAR bit affects the presentation of the ADC conversion result in the ADC Data Register. Write one to ADLAR to left adjust the result. Otherwise, the result is right adjusted. Changing the ADLAR bit will affect the ADC Data Register immediately, regardless of any ongoing conversions. For a complete description of this bit, see “The ADC Data Register – ADCL and ADCH”.

iii) Bits 4:0 – MUX4:0: Analog Channel and Gain Selection Bits

The value of these bits selects which combination of analog inputs are connected to the ADC. These bits also select the gain for the differential channels. See Table 98 for details. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set).

4.6.2. ADC Control and Status Register A – ADCSRA

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

i) Bit 7 – ADEN: ADC Enable

Writing this bit to one enables the ADC. By writing it to zero, the ADC is turned off. Turn- ing the ADC off while a conversion is in progress, will terminate this conversion.

ii) Bit 6 – ADSC: ADC Start Conversion

In Single Conversion mode, write this bit to one to start each conversion. In Free Running mode, write this bit to one to start the first conversion. The first conversion after ADSC has been written after the ADC has been enabled, or if ADSC is written at the same time as the ADC is enabled, will take 25 ADC clock cycles instead of the normal 13. This first conversion performs initialization of the ADC. ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect.

iii) Bit 5 – ADFR: ADC Free Running Select

When this bit is written to one, the ADC operates in Free Running mode. In this mode, the ADC samples and updates the data registers continuously. Writing zero to this bit will terminate Free Running mode.

iv) Bit 4 – ADIF: ADC Interrupt Flag

This bit is set when an ADC conversion completes and the data registers are updated. The ADC Conversion Complete Interrupt is executed if the ADIE bit and the I-bit in SREG are set. ADIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ADIF is cleared by writing a logical one to the flag. Beware that if doing a read-modify-write on ADCSRA, a pending interrupt can be disabled. This also applies if the SBI and CBI instructions are used.

v) Bit 3 – ADIE: ADC Interrupt Enable

When this bit is written to one and the I-bit in SREG is set, the ADC Conversion Complete Interrupt is activated.

vi) Bits 2:0 – ADPS2:0: ADC Prescaler Select Bits

These bits determine the division factor between the XTAL frequency and the input clock to the ADC.

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Πίνακας 4.9 *ADC Prescaler Selections*

4.6.3 The ADC Data Register – ADCL and ADCH

ADLAR = 0:

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	ADC9	ADC8	ADCH
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

ADLAR = 1:

Bit	15	14	13	12	11	10	9	8	
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

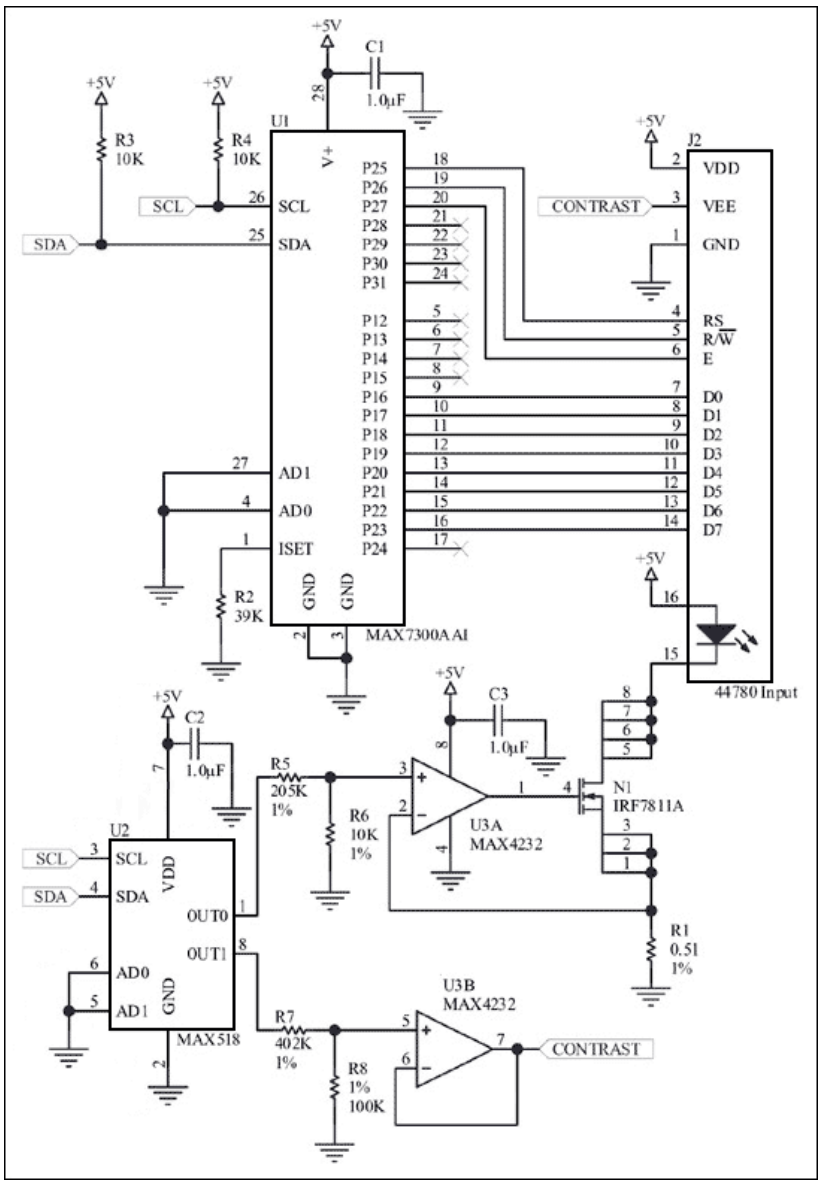
When an ADC conversion is complete, the result is found in these two registers. If differential channels are used, the result is presented in two's complement form. When ADCL is read, the ADC Data Register is not updated until ADCH is read. Consequently, if the result is left adjusted and no more than 8-bit precision is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, then ADCH. The ADLAR bit in ADMUX, and the MUXn bits in ADMUX affect the way the result is read from the registers. If ADLAR is set, the result is left adjusted. If ADLAR is cleared(default), the result is right adjusted.

ADC9:0: ADC Conversion Result

These bits represent the result from the conversion, as detailed in "ADC Conversion Result".

4.7. Οθόνη Υγρών Κρυστάλλων

Για την απεικόνιση των λειτουργιών της κατασκευής , χρησιμοποιείται η LCD HD44780 της Hitachi. Η HD44780 είναι μια οθόνη 4 γραμμών , με 20 χαρακτήρες ανά γραμμή.Στο παρακάτω σχήμα διακρίνεται το κύκλωμα της οθόνης.



PIN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
NAME	VSS	VDD	V0	RS	R/W	E	DB0	DB1	DB2	DB3	DB4	DB5	DB6	DB7	A	K

Πίνακας 4.10 *Λειτουργια Ακροδεκτών*

Vss	*GND*
Vdd	*Power Supply +5V*
Vo	*Liquid Crystal Driving Voltage*
RS	*Instruction Code Input : Data Input*
R/W	*Data Write from MPU to LCM*
E	*Enable*
DB0	*Data Bus Line*
DB1	*Data Bus Line*
DB2	*Data Bus Line*
DB3	*Data Bus Line*
DB4	*Data Bus Line*
DB5	*Data Bus Line*
DB6	*Data Bus Line*
DB7	*Data Bus Line*
Va	*Anode*
Vk	*Cathode*

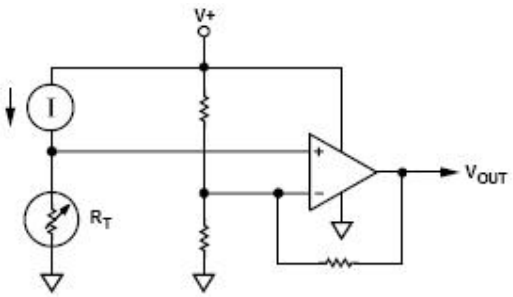
4.8. Αισθητήρια Θερμοκρασίας AD22100A

The AD22100 is a monolithic temperature sensor with on-chip signal conditioning. It can be operated over the temperature range −50°C to +150°C, making it ideal for use in numerous HVAC, instrumentation, and automotive applications.

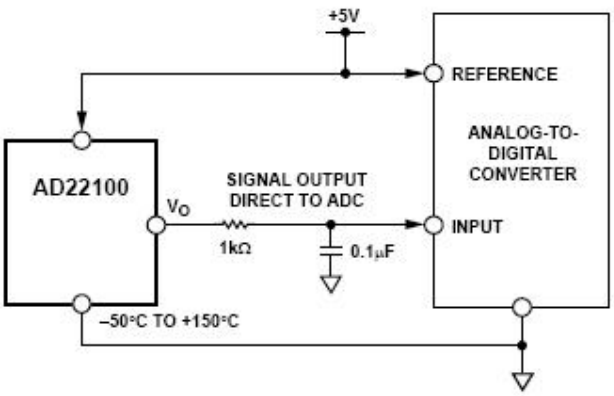
The signal conditioning eliminates the need for any trimming, buffering, or linearization circuitry, greatly simplifying the system design and reducing the overall system cost.

The output voltage is proportional to the temperature x the supply voltage (ratiometric). The output swings from 0.25 V at −50°C to +4.75 V at +150°C using a single +5.0 V supply.

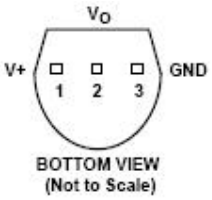
Due to its ratiometric nature, the AD22100 offers a cost-effective solution when interfacing to an analog-to-digital converter. This is accomplished by using the ADC's +5 V power supply as a reference to both the ADC and the AD22100 eliminating the need for and cost of a precision reference.



Σχήμα 4.11 Blog Diagram AD22100



Σχήμα 4.12 Κύκλωμα AD22100



Pin No.	Mnemonic	Description
1	V+	Power Supply Input.
2	Vo	Device Output.
3	GND	Ground Pin Must Be Connected to 0 V.

Σχήμα 4.13 Ακροδέκτες και η Λειτουργία τους

T_A = 25°C and V+ = 4 V to 6.5 V, unless otherwise noted.

	AD22100K			AD22100A			AD22100S			
Parameter	**Min**	**Typ**	**Max**	**Min**	**Typ**	**Max**	**Min**	**Typ**	**Max**	**Unit**
TRANSFER FUNCTION	$V_{OUT} = (V+/5\ V) \times [1.375\ V + (22.5\ mV/°C) \times T_A]$									V
TEMPERATURE COEFFICIENT	$(V+/5\ V) \times 22.5$									mV/°C
TOTAL ERROR										
Initial Error										
T_A = 25°C		±0.5	±2.0		±1.0	±2.0		±1.0	±2.0	°C
Error Overtemperature										
$T_A = T_{MIN}$		±0.75	±2.0		±2.0	±3.7		±3.0	±4.0	°C
$T_A = T_{MAX}$		±0.75	±2.0		±2.0	±3.0		±3.0	±4.0	°C
Nonlinearity										
$T_A = T_{MAX}$ to T_{MIN}			0.5			0.5			1.0	% FS[1]
OUTPUT CHARACTERISTICS										
Nominal Output Voltage										
V+ = 5.0 V, T_A = 0°C		1.375								V
V+ = 5.0 V, T_A = +100°C		3.625								V
V+ = 5.0 V, T_A = –40°C					0.475					V
V+ = 5.0 V, T_A = +85°C					3.288					V
V+ = 5.0 V, T_A = –50°C								0.250		V
V+ = 5.0 V, T_A = +150°C								4.750		V
POWER SUPPLY										
Operating Voltage	4.0	5.0	6.5	4.0	5.0	6.5	4.0	5.0	6.5	V
Quiescent Current		500	650		500	650		500	650	µA
TEMPERATURE RANGE										
Guaranteed Temperature Range	0		+100	–40		+85	–50		+150	°C
Operating Temperature Range	–50		+150	–50		+150	–50		+150	°C
PACKAGE		TO-92			TO-92			TO-92		
		SOIC			SOIC			SOIC		

Σχήμα 4.14 Χαρακτηριστικά του AD22100

The AD22100 is a ratiometric temperature sensor IC whose output voltage is proportional to its power supply voltage. The heart of the sensor is a proprietary temperature-dependent resistor, similar to an RTD, which is built into the IC. The temperature-dependent resistor, labeled R_T, exhibits a change in resistance that is nearly linearly proportional to temperature. This resistor is excited with a current source that is proportional to the power supply vo ltage. The resulting voltage across R_T is therefore both supply voltage proportional and line-arly varying with temperature. The remainder of the AD22100 consists of an op amp signal conditioning block that takes the voltage across

R_T and applies the proper gain and offset to achieve the following output voltage function:

$$V_{OUT} = (V+/5\ V) \times (1.375\ V + 22.5\ mV/°C \times T_A)$$

4.8.1. MICROPROCESSOR A/D INTERFACE ISSUES

The AD22100 is especially well suited to providing a low cost temperature measurement capability for microprocessor/ microcontroller based systems. Many inexpensive 8-bit micro-processors now offer an onboard 8-bit ADC capability at a modest cost premium. Total cost of ownership then becomes a function of the voltage reference and analog signal conditioning necessary to mate the analog sensor with the microprocessor ADC. The AD22100 can provide an ideal low cost system by eliminating the need for a precision voltage reference and any additional active components. The ratiometric nature of the AD22100 allows the microprocessor to use the same power supply as its ADC reference. Variations of hundreds of millivolts in the supply voltage have little effect as both the AD22100 and the ADC use the supply as their reference. The nominal AD22100 signal range of 0.25 V to 4.75 V (−50°C to +150°C) makes good use of the input range of a 0 V to 5 V ADC. A single resistor and capacitor are recommended to pro-vide immunity to the high speed charge dump glitches seen at many microprocessor ADC inputs.

An 8-bit ADC with a reference of 5 V will have a least significant bit (LSB) size of 5 V/256 = 19.5 mV. This corresponds to a nominal resolution of about 0.87°C.

4.8.2. RATIOMETRICITY CONSIDERATIONS

The AD22100 will operate with slightly better accuracy than that listed in the data sheet specifications if the power supply is held constant. This is because the AD22100's output voltage varies with both temperature and supply voltage, with some errors. The ideal transfer function describing output voltage is:

$$(V+/5\ V) \times (1.375\ V + 22.5\ mV/°C \times T_A)$$

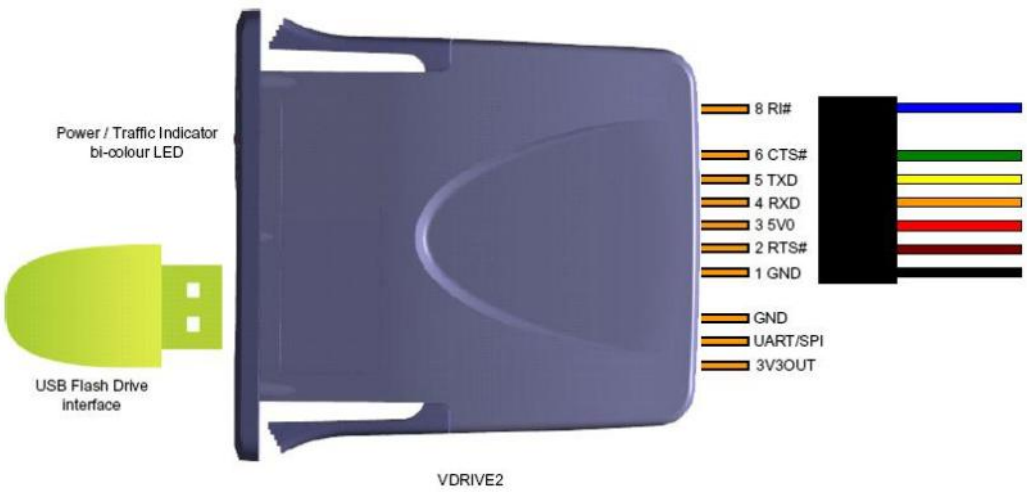
The ratiometricity error is defined as the percent change away from the ideal transfer function as the power supply voltage changes within the operating range of 4 V to 6.5 V. For the AD22100, this error is typically less than 1%. A movement from the ideal transfer function by 1% at 25°C, with a supply voltage varying from 5.0 V to 5.50 V, results in a 1.94 mV change in output voltage or 0.08°C error. This error term is greater at higher temperatures because the output (and error term) is directly proportional to temperature. At 150°C, the error in output voltage is 4.75 mV or 0.19°C.

For example, with V_S = 5.0 V, and T_A = +25°C, the nominal output of the AD22100 will be 1.9375 V. At V_S = 5.50 V, the nominal output will be 2.1313 V, an increase of 193.75 mV. A proportionality error of 1% is applied to the 193.75 mV, yielding an error term of 1.9375 mV. This error term translates to a variation in output voltage of 2.1293 V to 2.3332 V. A 1.94 mV error at the output is equivalent to about 0.08°C error in accuracy.

If 150°C is substituted for 25°C in the above example, the error term translates to a variation in output voltage of 5.2203 V to 5.2298 V. A 4.75 mV error at the output is equivalent to about 0.19°C error in accuracy.

4.9 USB Flash Device (VDRIVE)

The VDrive2 module provides an easy solution for adding a USB Flash disk interface to an existing product. Only four signal lines plus 5V supply and ground are required to be connected. Using the Vinculum VDAP firmware the VNC1L's I/O interface can be selected betwe en the serial UART or SPI using the on-board jumper pins. Not only is the VDrive2 ideal for evaluation and development of VNC1L designs, but also its neat enclosure and attractive quantity discount structure makes this module suitable for incorporation int o finished product designs. The VDrive2 is ideal for commercial products such as domestic goods, set top box, etc., as well as industrial products such as data loggers, software upgradable products, etc.



Εικόνα 4.15 VDrive2 Pin Out - UART interface.

UART/SPI	I/O Mode
Pull-Up	Serial UART
Pull-Down	SPI

Πίνακας 4.16 Port Selection Jumper Pins

UART Interface Signal Descriptions

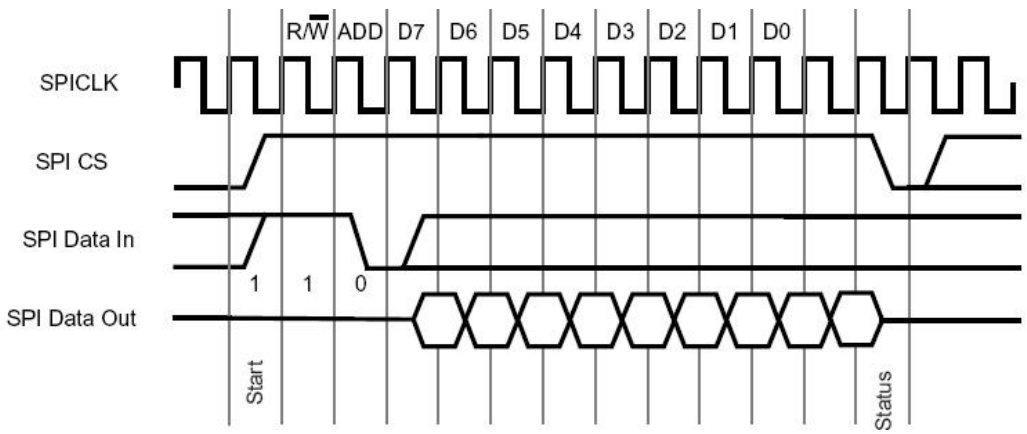
Pin No.	Name	Type	Description
1	GND	PWR	Signal ground
2	RTS#	Output	Request To Send Control Output / Handshake signal
3	5V0	PWR	5V supply input
4	RXD	Input	Receive asynchronous data input
5	TXD	Output	Transmit asynchronous data output
6	CTS#	Input	Clear To Send Control Input / Handshake signal
7	NC	-	No Connect
8	RI#	Input	Ring Indicator Control Input. Used to resume the Vinculum from suspend.

Πίνακας 4.17 Data and Control Bus Signal Mode Options - UART Interface.

SPI Interface Signal Descriptions and Timing Diagrams

Pin No.	Name	Type	Description
5	SCLK	Input	SPI Clock input, 12MHz maximum.
4	SDI	Input	SPI Serial Data Input
2	SDO	Output	SPI Serial Data Output
6	CS	Input	SPI Chip Select Input

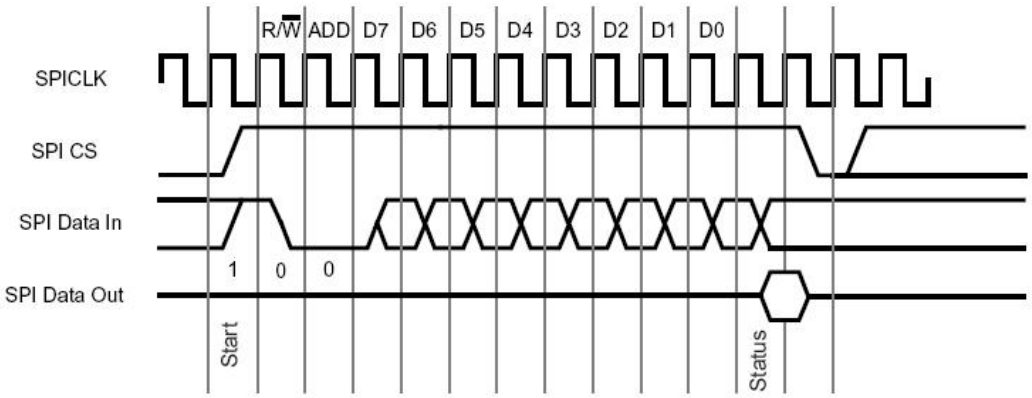
Πίνακας 4.18 Data and Control Bus Signal Mode Op tions - SPI Interface.



Διάγραμμα 4.19 SPI Slave Data Read Cycle.

From Start - SPI CS must be held high for the entire read cycle, and must be taken low for at least one clock period after the read is completed. The first bit on SPI Data In is the R/W bit - inputting a '1' here allows data to be read from the chip. The next bit is the address bit, ADD, which is used to indicate whether the data register ('0') or the status register ('1') is read from. During the SPI read cycle a byte of data will start being output on SPI Data Out on the next clock cycle after the address bit, MSB first.

After the data has been clocked out of the chip, the status of SPI Data Out should be checked to see if the data read is new data. A‘0’level here on SPI Data Out means that the data read is new data. A‘1’indicates that the data read is old data, and the read cycle should be repeated to get new data. Remember that CS must be held low for at least one clock period before being taken high again to continue with the next r ead or write cycle.



Διάγραμμα 4.20 SPI Slave Data Write Cycle.

From Start - SPI CS must be held high for the entire write cycle, and must be taken low for at least one clock period after the write is completed. The first bit on SPI Data In is the R/W bit - inputting a ‘0’ here allows data to be written to the chip. The next bit is the address bit, ADD, which is used to indicate whether the data register (‘0’) or the status register (‘1’) is written to. During the SPI write cycle a byte of data can be i nput to SPI Data In on the next clock cycle after the address bit, MSB first. After the data has been clocked in to the chip, the status of SPI Data Out should be checked to see if the data read was accepted. A‘0’level on SPI Data Out means that the data write was accepted. A‘1’indicates that the internal buffer is full, and the write should be repeated. Remember that CS must be held low for at least one clock period before being taken high again to continue with the next read or write cycle.

Κεφάλαιο 5

(Περιγραφή Λειτουργίας της Συσκευής)

5.1. Χρήση Συσκευής και Εντολές Λειτουργίας

5.1.1. Έναρξη Λειτουργίας

Η έναρξη της συσκευής γίνεται με τον διακόπτη Ο/Ι. Όταν ο διακόπτης πατηθεί στην θέση «Ι», η συσκευή τροφοδοτείται και η λειτουργία της ξεκινά. Η πρώτη ένδειξη που εμφανίζει η οθόνη είναι οι τίτλοι ενάρξεως της συσκευής.

	D	i	g	i	t	a	l		T	h	e	r	m	o	m	e	t	e	r
					L	o	a	d	i	n	g	.	.	.					

Στην συνέχεια ο μικροεπεξεργαστής εμφανίζει στην LCD οθόνη το κεντρικό menu της συσκευής. Στο κεντρικό menu της συσκευής μας εμφανίζονται όλες οι απαραίτητες μεταβλητές που πρέπει να ρυθμιστούν έτσι ώστε να ξεκινήσει το σύστημά μας τις μετρήσεις.

1	.	V	a	l	u	e	s					5	.	D	a	t	e	
2	.	R	a	n	g	e						6	.	M	o	d	e	
3	.	I	n	f	o							7	.	R	e	s	e	t
4	.	T	i	m	e							8	.	S	t	a	r	t

5.1.2. Επιλογή 1η – Values

Η πρώτη επιλογή που εμφανίζεται στην οθόνη είναι το υπομενού "Values". Μόλις πατήσουμε το "1" από το πληκτρολόγιό μας θα εισέλθουμε σε αυτό το υπομενού. Στο "Values" εμφανίζονται όλες οι ρυθμίσεις που έχουμε επιλέξει για το σύστημά μας (Ημερομηνία - Ώρα – Mode). Οι default τιμές των μεταβλητών μας (χωρίς να έχει γίνει ρύθμισή τους) είναι όλες "0". Πατώντας το "A" βγαίνουμε από το "Values" στο κεντρικό menu.

1	.	D	a	t	e	:	0	0	-	0	0	-	0	0				
2	.	T	i	m	e	:	0	0	:	0	0							
3	.	M	o	d	e	:	F	r	i	d	g	e						
A	.	E	x	i	t													

5.1.3. Επιλογή 2η – Range

Η δεύτερη επιλογή του κεντρικού μενού είναι το "Range" (δηλαδή η περιοχή μέτρησης θερμοκρασίας του συστήματ ός μας). Για να εισέλθουμε στο συγκεκριμένο υπομενού πατάμε το "2". Στο "Range" δεν μπορούμε να πειράξουμε κάποια ρύθμιση, απλώς μας εμφανίζει την περιοχή μέτρηση ς των αισθητηρίων μας που είναι -50^0 C – 150^0 C . Πατώντας το "A" βγαίνουμε από το "Range" στο κεντρικό menu.

M	a	x	i	m	u	m	:	+	1	5	0							
M	i	n	i	m	u	m	:	-	5	0								
A	.	E	x	i	t													

5.1.4. Επιλογή 3η – Info

Η τρίτη επιλογή του κεντρικού μενού είναι το "Info". Για να εισέλθουμε στο συγκεκριμένο υπομενού πατ άμε το "3". Στο "Info" εμφανίζεται η μονάδα

μέτρησης της θερμοκρασίας (Celsius) καθώς και ο χρόνος δειγματοληψίας των μετρήσεων (sampling rate=5sec). Πατώντας το "A" βγαίνουμε από το "Info" στο κεντρικό menu.

T	e	m	p	r	.		-	>		C	e	l	s	i	u	s		
F	r	i	d	g	e		-	>		P	e	r		5	s	e	c	.
R	o	o	m				-	>		P	e	r		1	0	s	e	C	.
A	.	E	x	i	t													

5.1.5. Επιλογή 4η – Time

Η τέταρτη επιλογή του κεντρικού μενού είναι το "Time". Για να εισέλθουμε στο συγκεκριμένο υπομενού πατάμε το "4". Στο "Time" σετάρουμε τον χρόνο έναρξης του συστήματός μας σε ώρες και λεπτά. Ο χρόνος αυτός αποθηκεύεται σε μια μεταβλητή "time" η οποία εμφανίζει τον χρόνο και στο υπομενού "Values". Πατώντας το "A" γίνεται αποθήκευση και έξοδος από το "Time" με επιστροφή στο κεντρικό menu.

S	e	t		T	i	m	e	:		0	0	:	0	0				
A	.	S	a	v	e		a	n	d		E	x	i	t				

5.1.6. Επιλογή 5η – Date

Η πέμπτη επιλογή του κεντρικού μενού είναι το "Date". Για να εισέλθουμε στο συγκεκριμένο υπομενού πατάμε το "5". Στο " Date" σετάρουμε την ημερομηνία έναρξης του συστήματός μας για date ,month, year . Η ημερομηνία αυτή αποθηκεύεται σε μια μεταβλητή "date" η οποία εμφανίζει την ημερομηνία και στο υπομενού "Values". Πατώντας το "A" γίνεται αποθήκευση και έξοδος από το " Date" στο κεντρικό menu.

S	e	t		D	a	t	e	:		0	0	/	0	0	/	0	0		
A	.	S	a	v	e		a	n	d		E	x	i	t					

5.1.7. Επιλογή 6η – Reset

Η προτελευταία επιλογή του κεντρικού μενού είναι το "Reset". Για να εισέλθουμε στο συγκεκριμένο υπομενού πατάμε το "7" από το πληκτρολόγιό μας. Στο συγκεκριμένο υπομενού το μόνο που μπορούμε να κάνουμε είναι πατώντας το πλήκτο "B" μηδενίζονται όλες οι μεταβλητές που έχουμε σετάρει στο σύστημά μας. Μόλις πατήσουμε το "B" γίνεται αυτόματη έξοδος στο κεντρικό μενού του συστήματός μας και μπορούμε να επαναεισάγουμε τιμές.

					P	r	e	s	s		‘	B	‘					
		D	e	f	a	u	l	t		V	a	l	u	e	s			
						&		E	x	i	t							

5.1.8. Επιλογή 7η – Start

Η τελευταία επιλογή του κεντρικού μενού μας είναι το "Start". Για να ξεκινήσει το σύστημά μας τις μετρήσεις, πρέπει να πατήσουμε το "8". Μόλις ξεκινήσει το σύστημά μας εμφανίζονται στην οθόνη οι τέσσε ρις θερμοκρασίες που αντιστοιχούν στα 4 αισθητήρια.

	T	1	:	2	8	,	5	2			T	2	:	2	8	,	2	7	
	T	3	:	2	8	,	3	0			T	4	:	2	8	,	2	7	

5.2. Βασική Λειτουργία της Συσκευής

Η συσκευή αποτελείται από 4 αισθητήρια AD22100 έτσι ώστε να μετράμε ταυτόχρονα τέσσερις διαφορετικές θερμοκρασίες με ακρίβεια δυο δεκαδικών ψηφίων (ή $0,01^0$ C). Με την βοήθεια της συσκευής Vdrive έχουμε την δυνατότητα να αποθηκεύουμε τις μετρήσεις μας σε μια USB Flash συσκευή. Η συσκευή αυτή μπορεί να είναι χωρητικότητας μέχρι 2 Gb διότι μεγαλύτερης χωρητικότητας USB Flash δεν αναγνωρίζεται από το Vdrive. Το σύστημά μας πρέπει να ξεκινάει εφόσον έχει τοποθετηθεί στο Vdrive η αποθηκευτική μας συσκευή γιατί σε οποιαδήποτε άλλη περίπτωση η κατασκευή δεν ανταποκρίνεται επειδή δεν υπάρχει κάποιο αποθηκευτικό μέσο για να εντοπίσει.

Ο αριθμός των μετρήσεων που παίρνει το σύστημα είναι προκαθορισμένος (65536) από τον κώδικα και μπορεί να αλλαχτεί. Επιπλέον ο χρόνος λήψης της κάθε μέτρησης καθορίζεται από τον κώδικα και είναι σταθερός (5 sec ανά μέτρηση σαν default) οπότε και η συνολική διάρκεια λειτουργίας της συσκευής (περίπου 91 ώρες). Σε περίπτωση που επιθυμούμε να αλλάξουμε τον χρόνο λήψης της κάθε μέτρησης μπορούμε εύκολα να το πετύχουμε αλλάζοντας απλώς τον timer που έχουμε επιλέξει μέσω κώδικα. Με τρόπο αυτό μπορούμε να αυξήσουμε ή να μειώσουμε τον χρόνο δειγματοληψίας. Αρχικά ο σχεδιασμός του συστήματός μας ήταν να έχει δυο διαφορετικά mode με διαφορετικό χρόνο δειγματοληψίας και αποθήκευσης των μετρήσεων. Με τον τρόπο αυτό θα μπορούσαμε να μειώσουμε ή και να αυξήσουμε τον όγκο των αποθηκευμένων δεδομένων.

Η ψηφιακή απεικόνιση της θερμοκρασίας έγινε με τον εξής τρόπο : Τα αισθητήρια AD22100 είναι στην ουσία μεταβλητές αντιστάσεις όπου η τιμή τους προκύπτει με βάση την εκάστοτε θερμοκρασία στον περιβάλλοντα χώρο. Η περιοχή μέτρησής τους είναι από -50 ^oC μέχρι +150 ^oC με σταθερή τροφοδοσία στα +5V. Η έξοδός τους είναι τάση, όπου στους -50 C^o αντιστοιχεί στα 0.25V και για τους +150 C^o έχουμε 4.75V . Στην ενδιάμεση περιοχή συμπεριφέρονται γραμμικά. Την μετατροπή του αναλογικού σήματος της εξόδου των αισθητηρίων σε ψηφιακή αναλαμβάνει ο ADC (Analog to Digital Converter) που είναι ενσωματωμένος στον ATmega128 με περιοχή λειτουργίας 0 - 5V. Αυτός περιέχει ένα 10-bit καταχωρητή τον ADCW (*βλ. 4.6. ADC Register Description)* όπου αποθηκεύεται η εκάστοτε τιμή σε ψηφιακή πλέον μορφή εύρους 10-bit δηλ. από 0 έως 1023.

Αφού ολοκληρωθεί η διαδικασία μέτρησης και αποθήκευσης στο USB Flash τότε έχει δημιουργηθεί ένα txt αρχείο με όνομα TIMES.txt το οποίο είναι αναγνώσιμο από οποιοδήποτε υπολογιστικό σύστημα. Αντιγράφοντας τα

δεδομένα από το USB Flash στον σκληρό δίσκο του υπολογιστή μας μπορούμε να τα επεξεργαστούμε με την βοήθεια του Excel της Microsoft.

Ακολουθεί ένα παράδειγμα με την μορφή που εμφανίζονται οι αποθηκευμένες μετρήσεις στον Η/Υ.

Date : 03 - 09 - 08

Time: 15:52

Channel1	Channel2	Channel3	Channel4
28.54	28.52	28.52	28.74
28.54	28.57	28.52	28.52
28.95	28.52	28.52	28.52
28.95	28.52	28.52	28.74
28.74	28.74	28.52	28.95
28.74	28.30	28.74	28.30
28.74	28.52	28.52	28.52
28.95	28.39	28.52	28.74
28.95	28.17	28.52	28.30
29.07	28.74	28.52	28.74

Όπου στο

Date : 03 είναι η ημέρα του μήνα που ξεκίνησε το σύστημα.

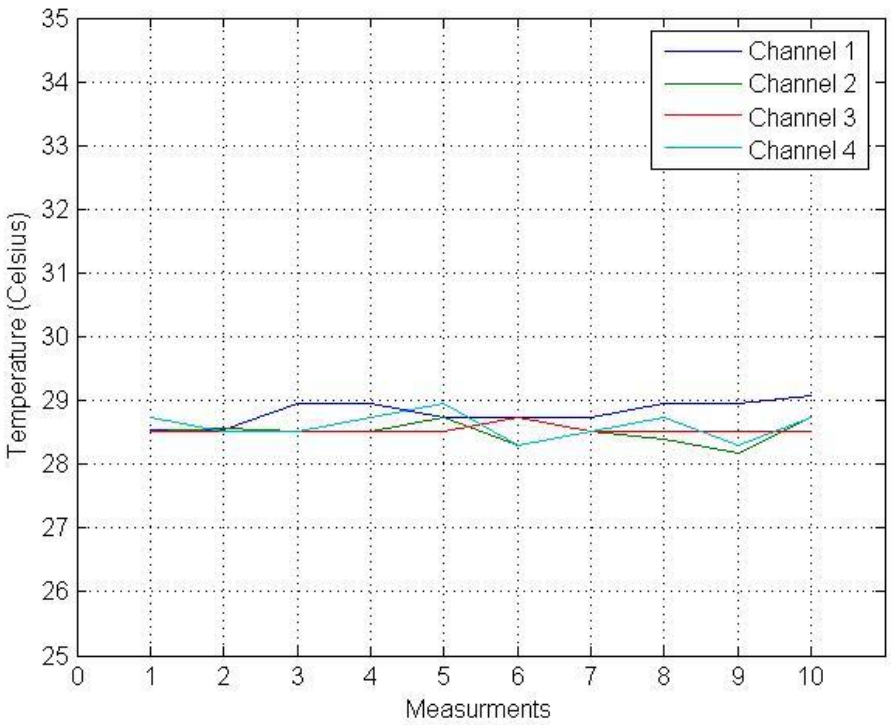
09 είναι ο μήνας του έτους που ξεκίνησε το σύστημα.

08 το έτος που ξεκίνησε το σύστημα.

Time : 15 η ώρα της ημέρας που ξεκίνησε το σύστημα.

52 τα λεπτά της ώρας που ξεκίνησε το σύστημα.

Στην συνέχεια εμφανίζονται τα κανάλια των μετρήσεων δηλαδή Channel 1,Channel 2,Channel 3 και Channel 4. Η κάθε στήλη κάτω από κάθε κανάλι αποτελεί τις μετρήσεις του κάθε καναλιού. Οι τιμές που εμφανίζονται στα 4 κανάλια είναι σε βαθμούς Κελσίου (^{o}C).

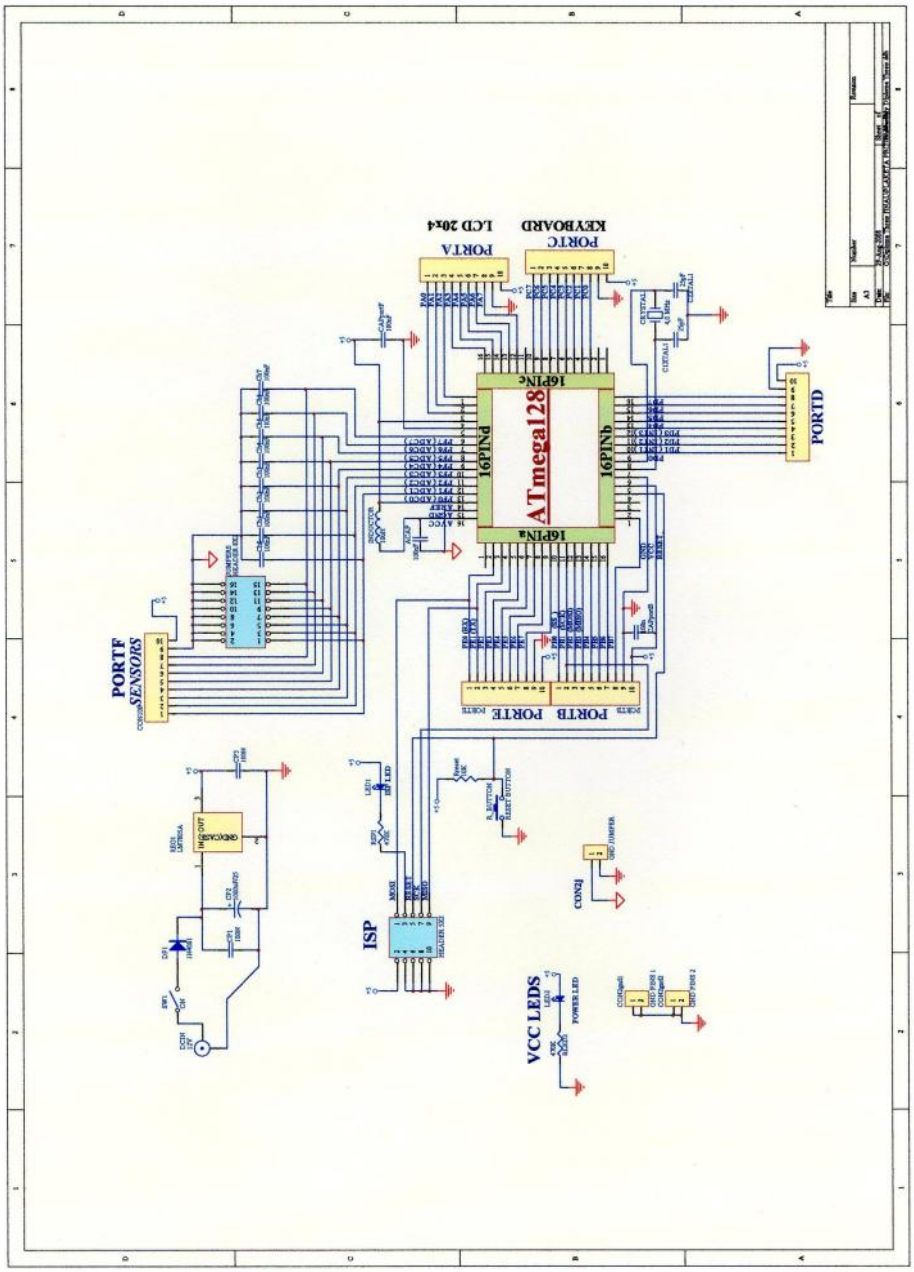


Το παραπάνω γράφημα παρουσιάζει τις αλλαγές της θερμοκρασίας των τεσσάρων αισθητηρίων σε 10 διαδοχικές μετρήσεις.

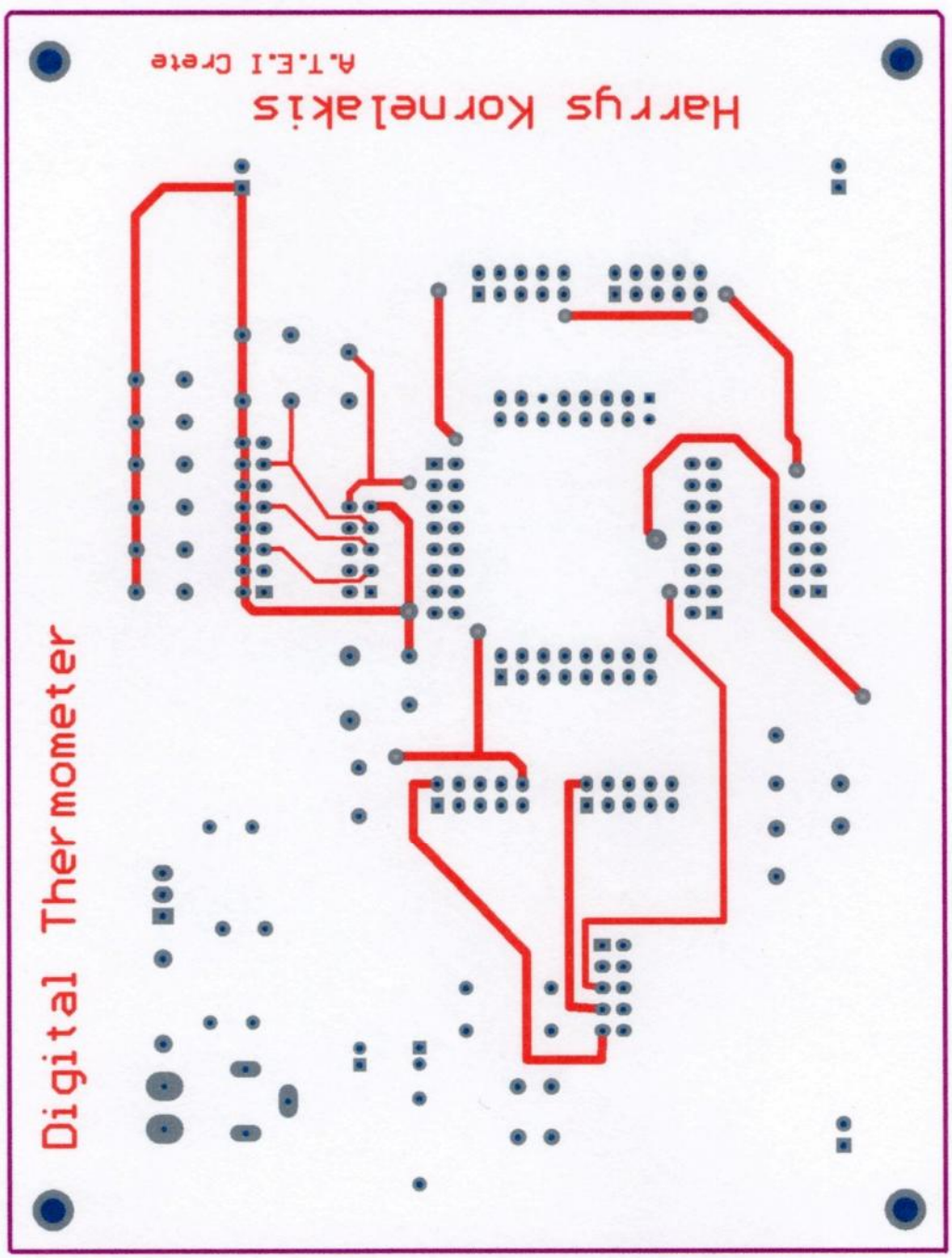
Κεφάλαιο 6

6.1.Schematics και Block Diagram

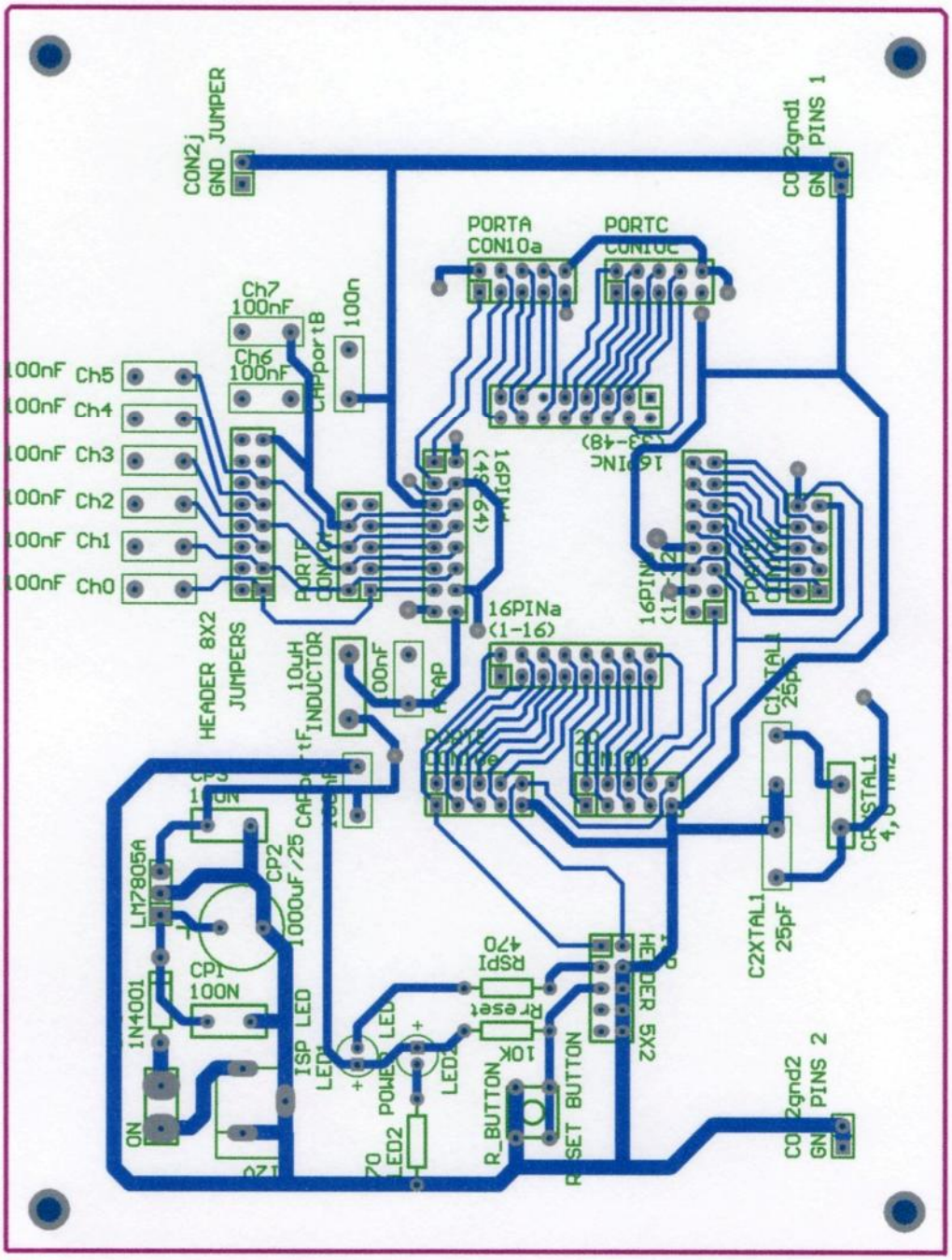
Παρακάτω εμφανίζονται τα σχηματικά της πλακέτας όπως κατασκευάστηκαν με την βοήθεια του Protel 99SE.



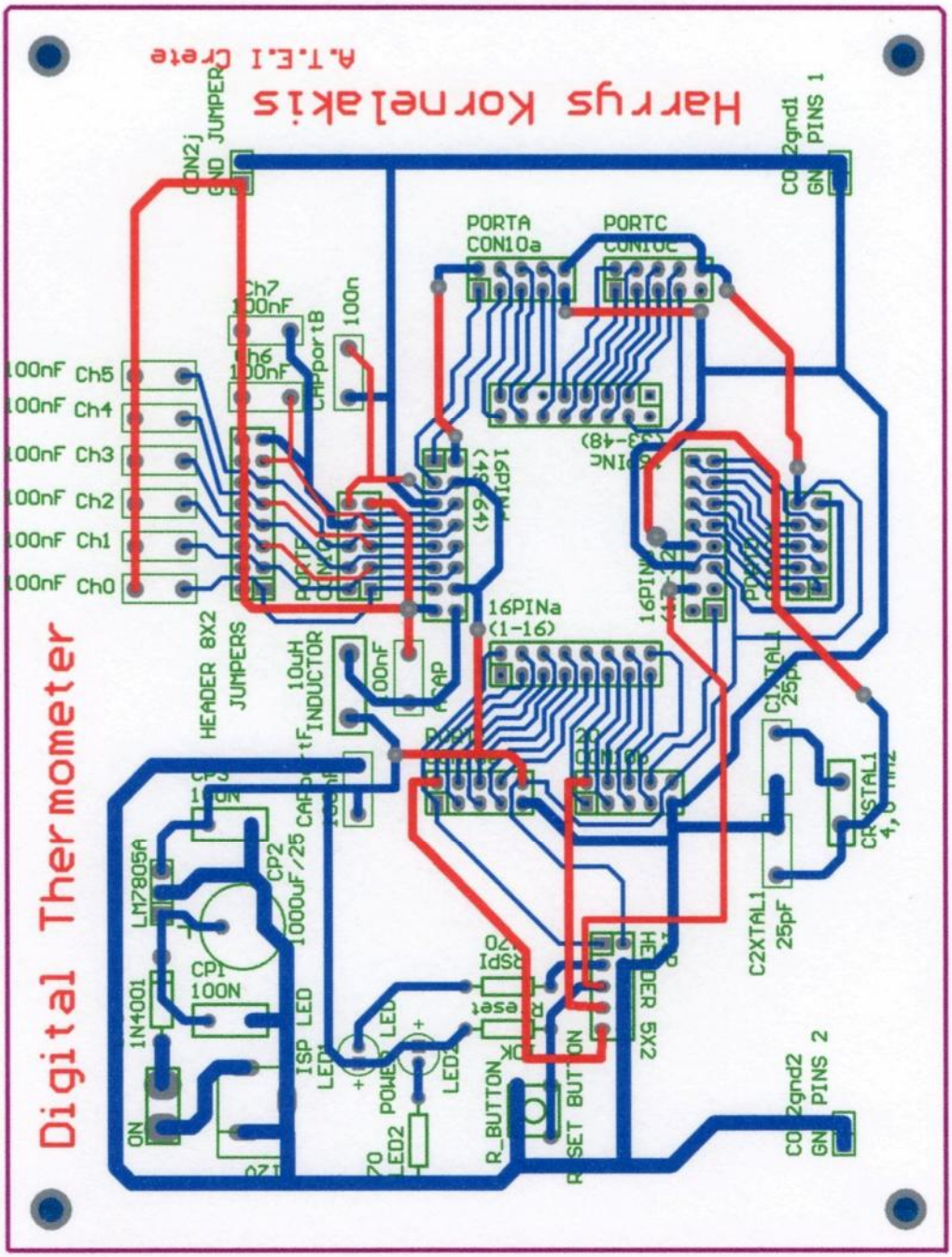
Σχήμα 1. Schematic Πλακέτας.



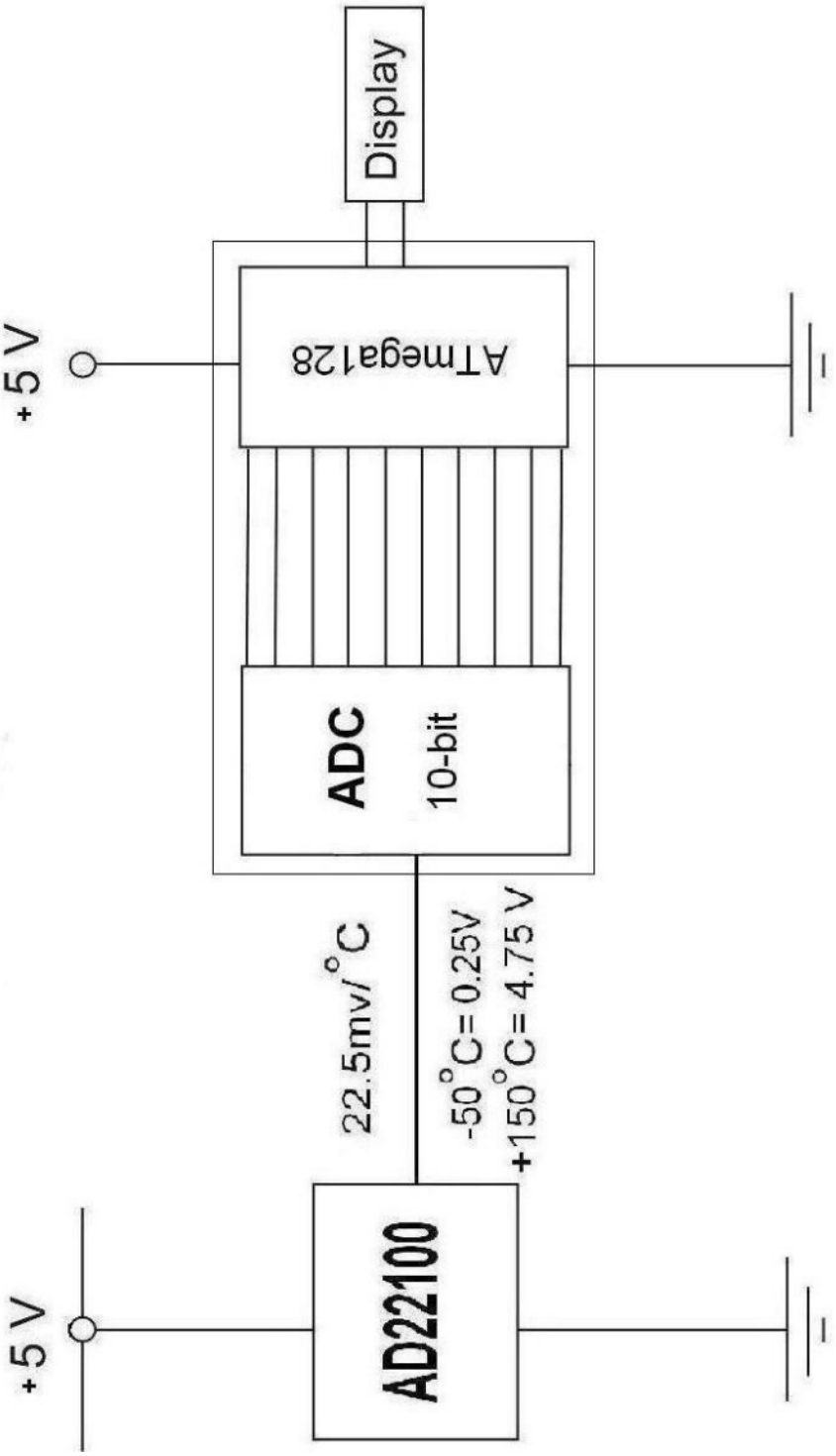
Σχήμα 2. Top Layer Πλακέτας.



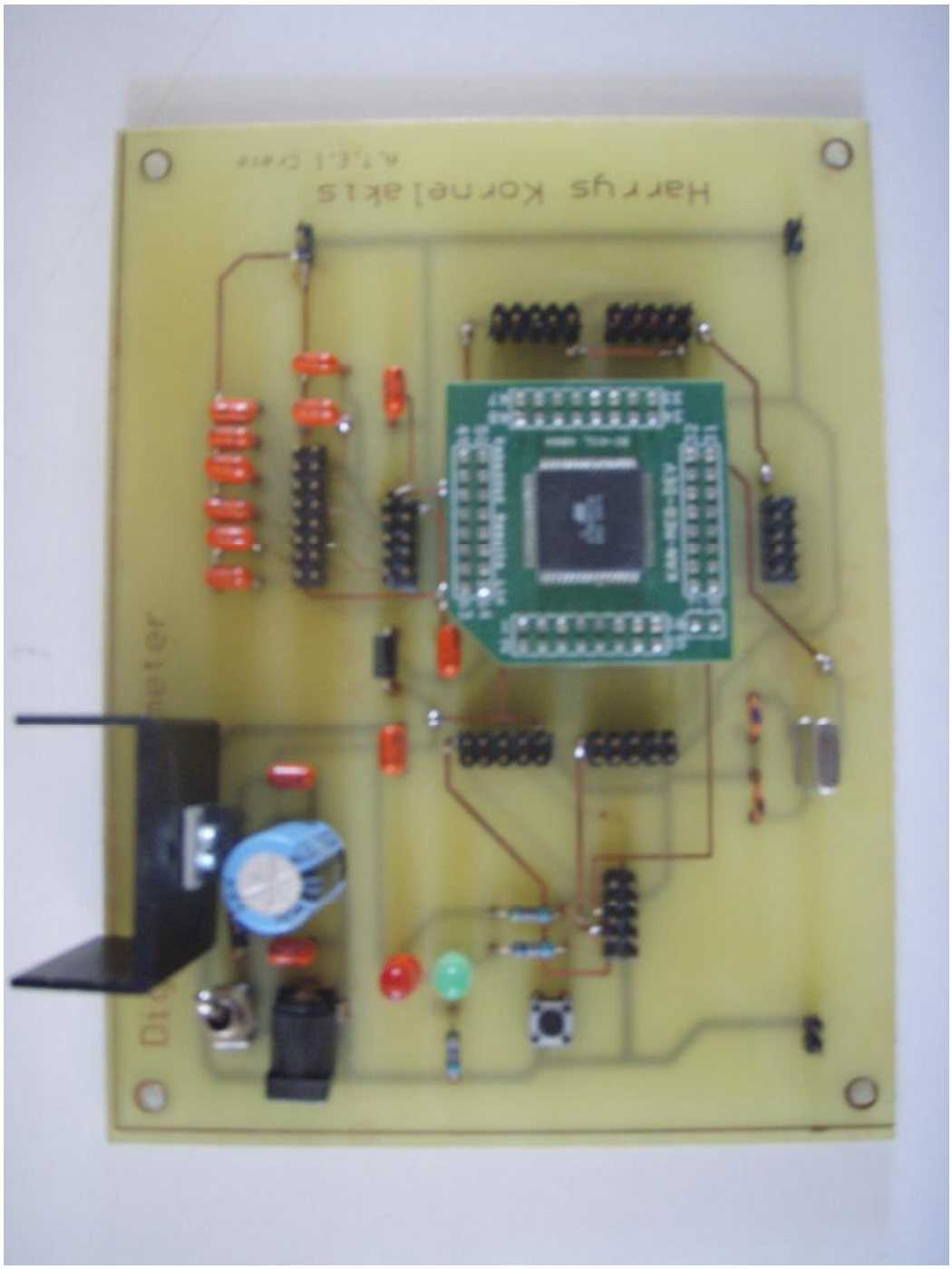
Σχήμα 3. Bottom Layer Πλακέτας.



Σχήμα 4. Bottom και Top Layer Πλακέτας.



Σχήμα 5. Block Diagram Συστήματος.



Σχήμα 6. Πλακέτα Κατασκευής.

ΠΑΡΑΡΤΗΜΑ

```
/*****************************************************
This program was produced by the
CodeWizardAVR V1.24.6 Standard
Automatic Program Generator
© Copyright 1998-2005 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com
e-mail:office@hpinfotech.com

Project :
Version :
Date    : 7/3/2008
Author  : John Minadakis
Company : T.E.I. of Crete
Comments:


Chip type           : ATmega128
Program type        : Application
Clock frequency     : 4.000000 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 1024
*****************************************************/

#include <mega128.h>
#include <delay.h>
```

```
//#include <stdio.h>

// Alphanumeric LCD Module functions
#asm
  .equ __lcd_port=0x1B ;PORTA
#endasm
#include <lcd.h>
#define Fridge 1
#define Room 2
#define gain 0.217

#define RXB8 1
#define TXB8 0
#define UPE 2
#define OVR 3
#define FE 4
#define UDRE 5
#define RXC 7

#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

// USART Receiver buffer
```

```
#define RX_BUFFER_SIZE 16
char rx_buffer[RX_BUFFER_SIZE];

#if RX_BUFFER_SIZE<256
unsigned char rx_wr_index,rx_rd_index,rx_counter;
#else
unsigned int rx_wr_index,rx_rd_index,rx_counter;
#endif

// This flag is set on USART Receiver buffer overflow
bit rx_buffer_overflow;

// USART Receiver interrupt service routine
interrupt [USART0_RXC] void usart0_rx_isr(void)
{
char status,data;
status=UCSR0A;
data=UDR0;
if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
  {
  rx_buffer[rx_wr_index]=data;
  if (++rx_wr_index == RX_BUFFER_SIZE) rx_wr_index=0;
  if (++rx_counter == RX_BUFFER_SIZE)
    {
    rx_counter=0;
    rx_buffer_overflow=1;
```

```
    };
  };
}

#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void)
{
char data;
while (rx_counter==0);
data=rx_buffer[rx_rd_index];
if (++rx_rd_index == RX_BUFFER_SIZE) rx_rd_index=0;
#asm("cli")
--rx_counter;
#asm("sei")
return data;
}
#pragma used-
#endif
// Standard Input/Output functions
#include <stdio.h>
#include "flashdisk.c"

unsigned int adc_data;
```

```
#define ADC_VREF_TYPE 0x00

// Ñïõôßíá interrupt ôïõ ADC

interrupt [ADC_INT] void adc_isr(void)
{
// Read the AD conversion result
adc_data=ADCW;   //ÁðïèÞêåõóç ôéìÞò ADCW óå ìåôáâëçôÞ adc_data
}

// ÁíÜãíùóç áðïôåëÝóìáôïò ADC
// êáé êáèÜñéóìá áðü èüñõâïï
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input | ADC_VREF_TYPE;
#asm
   in   r30,mcucr
   cbr  r30,__sm_mask
   sbr  r30,__se_bit | __sm_adc_noise_red
   out  mcucr,r30
   sleep
   cbr  r30,__se_bit
   out  mcucr,r30
#endasm
return adc_data;
```

```
}
 // Global ÄÞëùóç Ìåôáâëçôþí


unsigned char a,b,c,x,y;
unsigned char day,month,year,hours,minutes;
char key,oldkey=15000,ButtonPressed,tmp,tmp2,Mod=1;
float ch0,ch1,ch2,ch3;
char ADChannel=0;
bit write_data=0;


// ÄÞëùóç üëùí ôùí óõíáñôÞóåùí
// ðïõ ÷ñçóéìïðïéïýíôáé óôïí êþäéêá


void KeyDecoder(void);
void values(void);
void menu(void);
void clear(void);
void Range(void);
void Info(void);
void Time(void);
void Date(void);
//void Mode(void);
void Reset(void);
void lcdputnum (long int,char,char,char);
void putnum (long int,char,char,char);
void print_adc(void);
```

```
int length(int);

int ischneg(int);


unsigned int in1,out1;


// Ñïîõôßíá Overflow Interrupt Timer 1

// ðïîõ åíåñãïðïéåßôáé êáèÝ äåõôåñüëåðôï


interrupt [TIM1_OVF] void timer1_ovf_isr(void)

{

static unsigned char secs=0;

static unsigned int count=0;

TCNT1H=0x0B;

TCNT1L=0xDC;


secs++;                    //êÜèå äåõôåñüëåðôï áõîÜíåôáé ç ìåôáâëçôÞ secs
êáôÜ Ýíá.


 if (count==13)            // count åßíáé ç ì åôáâëçôÞ ðïõ ìáò êáèïñßæåé ôïí
áñéèìü ôùí ìåôñÞóåùí

                    // ðïõ èá ðÜñåé ôï óýóôçìÜ ìáò.Ç ìÝãéóôç ôéìÞ ðïõ ìðïñåß íá
ðÜñåé

                    // ç ìåôáâëçôÞ count åßíáé count=65536.

      {


    write_data=0;          // Ìüëéò ôï count ãßíåé ßóï ìå ôçí ôéìÞ ðïõ ôïõ Ý÷ïõìå
```

```
    count=0;                // êáèïñßóåé ìçäáíßæåôáé ôïí write_data, ôï count
êëåßíáé

    VNC1L_CloseFile();      // ôï áñ÷åßï .txt êáé îáíáêáëïýìå ôçí óõíÜñôçóç
menu.

    menu();


      }


 if(secs == 5) {            // ÊÜèå 5 äåõôåñüëåðôá  ãßíåôáé ôï write_data=1
áõîÜíåôáé

                        // ôï count êáôÜ Ýíá êáé ìüëéò ôï secs ãßíåé ßóï ìå 5

    secs=0;             // ôï secs ìçäåíßæåé ãéá íá îåêéíÞóåé áðü ôçí áñ÷Þ.

    write_data=1;

    count++;

      }


    }


// ÓõíÜñôçóç setup.ÓåôÜñéóìá ôïõ ìéêñïåëåãêôÞ ìå ôïí êáèïñéóìü êáé ôçí
äÞëùóç ôùí ðåñéöåñåéáêþí ôïõ.

// ÓåôÜñéóìá Timer 1 , ADC êáé USART0 êáè þò êáé äÞëùóç ôùí ðïñôþí óáí
åéóüäïõò êáé åîüäïõò.


void setup(void){


// ÄÞëùóç Åéóüäùí/Åîüäùí


// Port A initialization
```

```
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In

// State7=T State6=T State5=T State4=T State3=T State2=T S tate1=T State0=T

PORTA=0x00;

DDRA=0x00;


// Port B initialization

// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In

// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T

PORTB=0x00;

DDRB=0xff;


// Port C initialization

// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In

// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T

PORTC=0x00;

DDRC=0x00;


// Port D initialization

// Func7=In Func6=In Func5=In Func4=In Func 3=In Func2=In Func1=In
Func0=In

// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T

PORTD=0x00;

DDRD=0x00;


// Port E initialization
```

```
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In

// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T

PORTE=0x00;

DDRE=0x00;


// Port F initialization

// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In

// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T

PORTF=0x00;

DDRF=0x00;


// Port G initialization

// Func4=In Func3=In Func2=In Func1=In Func0=In

// State4=T State3=T State2=T State1=T State0=T

PORTG=0x00;

DDRG=0x00;


// Timer/Counter 1 initialization

// Clock source: System Clock

// Clock value: 62.500 kHz

// Mode: Normal top=FFFFh

// OC1A output: Discon.

// OC1B output: Discon.

// OC1C output: Discon.

// Noise Canceler: Off
```

```
// Input Capture on Falling Edge

// Timer 1 Overflow Interrupt: On
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x03;
TCNT1H=0x0B;
TCNT1L=0xDC;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
OCR1CH=0x00;
OCR1CL=0x00;

// ÃÞëùóç Timer/Counter Interrupt
TIMSK=0x04;
ETIMSK=0x00;

// ÄÞëùóç USART0
// ÐáñÜìåôñïé Åðéêïéíùíßáò : 8 Data, 1 Stop, No Parity
```

```
// USART0 Receiver: On
// USART0 Transmitter: On
// USART0 Mode: Asynchronous
// USART0 Baud rate: 9600
UCSR0A=0x00;
UCSR0B=0x98;
UCSR0C=0x06;
UBRR0H=0x00;
UBRR0L=0x19;

// ÄÞëùóç Analog Comparator
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;


ADMUX=ADC_VREF_TYPE;
ADCSRA=0x8F;

// LCD module initialization
lcd_init(20);

// Global enable interrupts
#asm("sei")
```

```
 }


// ÓõíÜñôçóç ãéá ôçí åíåñãïðïßçóç ôïõ ADC


void extraSetup(void){

// ADC initialization

// ADC Clock frequency: 125.000 kHz

// ADC Voltage Reference: Int., cap. on AREF

// ADC High Speed Mode: Off

ADCSRA|=0xC8;   // Åíåñãïðïßçóç ôïõ Interrupt ôïõ ADC.

SREG|=0x80;     // ÃåíéêÞ åíåñãïðïßçóç Interrupt

SFIOR&=0xEF;


}


// ÓõíÜñôçóç main.Káëåßôáé ìüëéò áíïßîåé ôï óýóôçìá êáëåß ôçí setup,
extrasetup

// êáé åìöáíßæåé ôï ìÞíõìá "Digital Thermometer" óôçí LCD ãéá 3 seconds.

// ÌåôÜ êÜíåé clear óôçí ïèüíç êáé êáëåß ôçí óõíÜñôçóç ôïõ menu.


void main(void){


setup();

extraSetup();

lcd_putsf(" Digital Thermometer\n\n     Loading...");

delay_ms(3000);
```

```
putsf("Loading");



  clear();




while (1)
    {

     menu();

  };
}

// ÓõíÜñôçóç clear ç ïðïßá üðïôå êáëåßôáé êáèáñßæåé ôçí ïèüíç ôïõ L CD.
// Ç ëåéôïõñãßá ôçò åßíáé íá ðçãáßíåé êáé íá ãñÜöåé óå üëåò ôéò
ãñáììÝò
// êáé ôéò óôÞëåò ôçò LCD (20 óôÞëåò x êáé 4 ãñáììåò y) Ýíá êåíü.


  void clear(void){
 int x,y;


  for(y=0;y<4;y++){
          for(x=0;x<20;x++){
          lcd_gotoxy(x,y);
```

```
          lcd_putsf(" ");
          }
  }
    }
```

//ÓõíÜñôçóç KeyDecoder.ÁíÜãíùóç êáé êùäéêïðïßçóç ôïõ ðëçêôñïëïãßïõ 4x4.

```
void KeyDecoder (void){

    DDRC=0xff;
    PORTC=0x0f;
    DDRC=0x00;
    a=PINC;
    DDRC=0xff;
    PORTC=0xf0;
    DDRC=0x00;
    b=PINC;
    c=a&b;

 // ÁíÜëõóç ôùí button ôïõ ðëçêôñïëïãßïõ ìå âÜóç ôçí ìåôáâëçôÞ c.

      if(c==17){
         key=1;
      }else  if(c==18){
        key=2;
```

```
}else if(c==20){
  key=3;
}else if(c==24){
  key=10;
}else if(c==33){
  key=4;
}else if(c==34){
  key=5;
}else if(c==36){
  key=6;
}else if(c==40){
  key=11;
}else if(c==65){
  key=7;
}else if(c==66){
  key=8;
}else if(c==68){
  key=9;
}else if(c==72){
  key=12;
}else if(c==130){
  key=0;
}else if(c==129){
  key=15;
}else if(c==132){
  key=14;
```

```
      }else if(c==136){
        key=13;
      }else key=20;

 // To ÂuttonPressed åßíáé ìéá ìåôáâëçôÞ ðïõ ìáò êáèïñßœåé
 // ðüôå Ý÷åé ðáôçèåß Ýíá ðëÞêôñï.Ìüëéò ðáôÞóïõìå Ýíá ðëÞêôñï
 // ôï key ðáßñíåé ôçí áíôßóôïé÷ç ôéìÞ êáé ôï ÂuttonPressed ìçäåíßœåôáé.

      ButtonPressed=0;
      if(key!=oldkey || key==20){
       ButtonPressed=1;
       oldkey=key;
      }
    }

// ÓõíÜñôçóç êåíôñéêïý menu ôçò óõóêåõÞò.Ìüëéò ìðïýìå óôçí óõíÜñôçóç
// áõôÞ êáëåßôáé ç clear ãéá íá ãßíåé êáèáñéóìüò ôï õ LCD.ÁíÜëïãá ìå ôï
//ðëÞêôñï ðïõ èá ðáôÞóïõìå ìåôÜ ìðáßíïõìå óôï áíôßóôïé÷ï õðïìåíïý.

void menu(void){

       clear();


   while(2){
      lcd_gotoxy(0,0);
```

```
      lcd_putsf("1.Values     5.Date \n2.Range       6.Reset\n3.Info
7.Start\n4.Time");


      KeyDecoder();



 if(key==1&ButtonPressed==1){                  //Ðáôþíôáò ôï "1" ìðáßíïõìå óôçí
"values".
        ButtonPressed=0;
        values();
      }


      else if (key==2&ButtonPressed==1){      //Ðáôþíôáò ôï "2" ìðáßíïõìå óôçí
"range".
         ButtonPressed=0;
         Range();
      }


      else if(key==3&ButtonPressed==1){       //Ðáôþíôáò ôï "3" ìðáßíïõìå óô çí
"Info".
           ButtonPressed=0;
           Info();
        }


      else if(key==4&ButtonPressed==1){       //Ðáôþíôáò ôï "4" ìðáßíïõìå óôçí
"Time".
           ButtonPressed=0;
           Time();
```

```
        }


    else if(key==5&ButtonPressed==1){      //Ðáôþíôáò ôï "5" ìðáßíïõìå óôçí
"Date".
            ButtonPressed=0;
            Date();
        }



    else if(key==6&ButtonPressed==1){      //Ðáôþíôáò ôï "7" ìðáßíïõìå óôçí
"Reset".
            ButtonPressed=0;
            Reset();
        }



    if(key==7 & ButtonPressed==1){         //Ðáôþíôáò ôï "8" ìðáßíïõìå óôçí
"print_adc".
            ButtonPressed=0;


            print_adc();
            break;
    }else ;


  };
}
```

```
// ÓõíÜñôçóç values.Êáëåßôáé ðñþôç ç clear êáé ìáôÜ ôõðþíåé óôçí LCD óôéò

// èÝóåéò ðïõ ôïõ Ý÷ïõìå äþóåé ôï Date ôï Time êáé ôï Mode.Åðßóçò êáëïýìå

// êáé ôçí KeyDecoder ãéá íá ìðïñ åß íá ãßíåôáé áíÜãíùóç ôïõ ðëçêôñïëïãßïõ.


void values(void){

      clear();

  lcd_gotoxy(0,0);

  lcd_putsf("1.Date:");

lcdputnum (day,2,'0',0);lcd_putchar('-');lcdputnum
(month,2,'0',0);lcd_putchar('-');lcdputnum(year,2,'0',0);

  lcd_gotoxy(0,1);

  lcd_putsf("2.Time:");

lcdputnum (hours,2,'0',0);lcd_putchar(':');lcdputnum (minutes,2,'0',0);

  lcd_gotoxy(0,3);

  lcd_putsf("A.Exit");


      while(2){

      KeyDecoder();

   if(key==10 && ButtonPressed==1){     // ÅÜí ôï key=10 äçë. ðáôÞóï õìå ôï "Á"
âãáßíïõìå óôï

     ButtonPressed=0;                 // óôï êåíôñéêü menu ôçò óõóêåõÞò ìáò.

       break;

         }

      };

      clear();


}
```

```
// ÓõíÜñôçóç Range.Êáëïýìå ôçí clear êáé ôçí KeyDecoder åíþ ôõðþíïõìå óôçí ïèüíç áðëþò ôï

// Maximum:+150,Minimum:-50 åíþ ôï ìüëéò ôï key=10 ãßíåôáé break êáé âãáßíáé áðü ôçí óõíÜñôçóç.


void Range(void){
   clear();
   lcd_gotoxy(0,0);
   lcd_putsf("Maximum:+150\nMinimum:-50\n\nA.Exit");
   while(2){
      KeyDecoder();
      if(key==10 & ButtonPressed==1){
           ButtonPressed=0;
           break;
        }
   };
   clear();

}


// ÓõíÜñôçóç Info.Êáëïýìå ôçí clear êáé ôçí KeyDecoder åíþ ôõðþíïõìå óôçí ïèüíç áðëþò ôçí ïííÜáá

// ìÝôñçóçò ôçò èåñìïêñáóßáò êáé üôáí  ôï key=10 ãßíåôáé break êáé âãáßíáé áðü ôçí óõíÜñôçóç.


void Info(void){
   clear();
```

```
   lcd_gotoxy(0,0);
   lcd_putsf("Tempr. -> Celsius\nFridge -> Per 5sec.\nRoom   -> Per
10sec.\nA.Exit");
   while(2){
      KeyDecoder();
      if(key==10 & ButtonPressed==1){
           ButtonPressed=0;
           break;
        }
   };
   clear();
}


// ÓõíÜñôçóç Time.ÃñÜöïõìå êáé óåôÜñïõìå ôçí þñá Ýíáñîçò ôïõ óõóôÞìáôüò
ìáò.
// Ç þñá åßóáãåôáé óôï óýóôçìá ìå ôçí ïïñöÞ 23:00.Ôá äõï ðñþôá øçößá
áíôéóôïé÷ïýí
// óôéò þñåò åíþ ôá äõï ôåëåõôáßá óôá ëåðôÜ.ÕðÜñ÷åé ðåñéïñéóìüò
Ýôóé þóôå íá ìçí
// ìðïñïýìå íá åéóÜãïõìå ëÜèïò áñéèìü äçë. ï áñéèìüò ôùí ùñþí íá ìçí åßíáé
ìåãáëýôåñïò
// áðï ôï 23 (hours=<23).Ìïíßùò óåôÜñáôáé êáé ï áñéèìüò ôùí ëåðôþí
(minutes=<59).


void Time(void){
   int count=0;
   ButtonPressed=0;
   clear();
   lcd_gotoxy(0,0);
```

```
   lcd_putsf("Set Time: __:__ \n\n\nA.Save and Exit");
   minutes=0;
   hours=0;
   count=0;                              // count åßíáé ç ìåôáâëçôÞ ðïõ áõîÜíåé êáôá
Ýíá
   while(count<2){                       // êÜèå öïñÜ ðïõ ðáôÜìå êÜðïéï ðëÞêôñï.
      KeyDecoder();
      if((ButtonPressed==1) & (key>=0 & key<10) ){
         ButtonPressed=0;
         tmp=(int)(key);
         hours=(hours*10)+tmp;
         lcd_gotoxy(10+count,0);
         lcd_putchar(key+48);
         count++;
      }
      if(hours>23){                  //Áí ôï hours>23  ôüôå ôï count=0 êáé ðÜåé êáé
ãñÜöåé
         count=0;                  // óôï LCD óôçí èÝóç (10+count,0) äõï êÜèåôåò
ðáýëåò
         hours=0;                 // Ýôóé þóôå íá óåôÜñåéò óùóôÜ ôçí þñá
         delay_ms(500);
         lcd_gotoxy(10+count,0);
         lcd_putsf("__");

      }
   }
   count=3;
   while(count<5){
```

```
      KeyDecoder();
      if((ButtonPressed==1) & (key>=0 & key<10)){
         ButtonPressed=0;
         tmp=(int)(key);
         minutes=(minutes*10)+tmp;
         lcd_gotoxy(10+count,0);
         lcd_putchar(key+48);
         count++;
      }
      if(minutes>59){             //Ái ôá minutes>59 ôüôå ôï count ãßíåôáé 3 êáé
ðÜåé êáé ãñÜöåé
         count=3;                // óôï LCD óôçí èÝóç (10+count,0) äõï êÜèåôåò
ðáýëåò Ýôóé þóôå
         minutes=0;              // íá óåôÜñåéò óùóôÜ ôçí þñá.
         delay_ms(500);
         lcd_gotoxy(10+count,0);
         lcd_putsf("__");

      }
   }

   while(1){                          //Êáëïýìå ôçí KeyDecoder  Ýôóé þóôå íá
åßíáé
      KeyDecoder();                   // äõíáôÞ ç ÷ñÞóç ôïõ ðëçêôñïëïãßïõ.
      if(key==10 & ButtonPressed==1){
        ButtonPressed=0;
          break;
      }
```

```
    }
  clear();
}


// ÓõíÜñôçóç Date .ÃñÜøïõìå êáé óåôÜñïõìå ôçí çìåñïìçíßá Ýíáñîçò ôïõ
óõóôÞìáôüò ìáò.
// Ç çìåñïìçíßá åßóáãåôáé óôï óýóôçìá ìå ôçí ïïñöÞ 25/08/08.Ôá äõï ðñþôá
øçößá áíôéóôïé÷ïýí
// óôçí çìÝñá ôá äõï åðüìåíá óôïí ìÞíá åíþ ôá ôåëåõôáßá äõï óôï Ýôïò Ýíáñîçò
ôïõ óôõóôÞìáôïò.
// ÕðÜñ÷åé ðåñéïñéóìüò Ýôóé þóôå íá ìçí ìðïñïýì íá åéóÜãïõìå ëÜèïò áñéèìü
äçë. ï áñéèìüò
// çìåñþí íá ìçí åßíáé ìåãáëýôåñïò áðï ôï 31 (day=<31).Ïìïßùò óåôÜñïíôáé êáé
ïé ìåôáâëçôÝò ôïõ
// ìÞíá (month=<12) êáé ôïõ Ýôïõò (year=<99).


void Date(void){
    int count=0;
   ButtonPressed=0;
   clear();
   lcd_gotoxy(0,0);
   lcd_putsf("Set Date: __/__/__ \n\n\nA.Save and Exit");
   day=0;
   month=0;
   year=0;                        // Ç ìåôáâëçôÞ count áõîÜíåé êáôÜ Ýíá êÜèå öïñÜ
ðïõ
    while(count<2){               // ðáôÜìå êÜðïéï ðëÞêôñï.
      KeyDecoder();
      if((ButtonPressed==1) & (key>=0 & key<10)){
```

```
        ButtonPressed=0;
        tmp2=(int)(key);
        day=(day*10)+tmp2;
        lcd_gotoxy(10+count,0);
        lcd_putchar(key+48);
        count++;
    }
    if((day>31||day==0) && count==2){
        count=0;
        day=0;
        delay_ms(500);
        lcd_gotoxy(10+count,0);
        lcd_putsf("__");

    }
}
count=3;
while(count<5){
    KeyDecoder();
    if((ButtonPressed==1) & (key>=0 & key<10)){
        ButtonPressed=0;
        tmp2=(int)(key);
        month=(month*10)+tmp2;
        lcd_gotoxy(10+count,0);
        lcd_putchar(key+48);
        count++;
```

```
    }
    if((month>12||month==0) && count==5){
        count=3;
        month=0;
        delay_ms(500);
        lcd_gotoxy(10+count,0);
        lcd_putsf("__");
    }
  }

    count=6;
  while(count<8){
    KeyDecoder();
    if((ButtonPressed==1) & (key>=0 & key<10)){
        ButtonPressed=0;
        tmp2=(int)(key);
        year=(year*10)+tmp2;
        lcd_gotoxy(10+count,0);
        lcd_putchar(key+48);
        count++;
    }
    if(year>99){
        count=6;
        year=0;
        delay_ms(500);
        lcd_gotoxy(10+count,0);
```

```
            lcd_putsf("__");

        }


    }


    while(1){
      KeyDecoder();
      if(key==10 & ButtonPressed==1){
           ButtonPressed=0;
           break;
      }
    }
  clear();
}


void Reset(void){
  clear();
     lcd_gotoxy(0,0);
     lcd_putsf("     Press 'B'\n  Default Values\n      & Exit");

       while(2){
      KeyDecoder();
```

```
  if(key==11 & ButtonPressed==1){
          ButtonPressed=0;
           hours=0;
           minutes=0;
           day=0;
           month=0;
           year=0;
           break;
        }
   };
   clear();
}

// ÓõíÜñôçóç putnum.Ï ñüëïò ôçò óõíÜñôçóçò áõôÞò åßíáé íá
// ôõðþíåé ôïõò integers óôçí ïèüíç ôïõ LCD.

void putnum (long int n,char c,char fill,char dp){
 char *num1="_________";
 char *p;
 long m,s;
 int i,flag;

 p=num1;
 i=10;                  //places
 s=1000000000;          //devisor
 flag=1;                //number found flag
```

```
  while(i--)
  {
    m=n/s;                //get next msd
    *p=(char)(m+48);      //convert to ascii
    if(flag && *p=='0' &&i>0)   //zap if leading zero
        *p=fill;
    else
       flag=0;            //next char
    p++;                  //shift left
    n=n-m*s;              // reduce divisor
    s/=10;
     }

p=num1+10-c;

for (i=10-c;i<10;i++) {
   if ((dp!=0) && (i==(10-dp))) putchar(',');
   putchar(*p);
   delay_ms(10);
   p++;

   }
}

// ÓõíÜñôçóç lcdputnum.Ï ñüëïò ôçò  óõíÜñôçóçò áõôÞò åßíáé íá
```

```
// ôõðþíåé ôïõò integers óõãêåêñéìÝíá áðü ôçí print_adc (äçë.ôéò
ìåôñÞóåéò ôïõ ADC)

// óôçí ïèüíç ôïõ LCD.


void lcdputnum (long int n,char c,char fill,char dp){
 char *num1="__________";
 char *p;
 long m,s;
 int i,flag;


 p=num1;
 i=10;                  //places
 s=1000000000;            //devisor
 flag=1;                  //number found flag


 while(i--)
 {
  m=n/s;                //get next msd
  *p=(char)(m+48);       //convert to ascii
  if(flag && *p=='0' &&i>0)   //zap if leading zero
     *p=fill;
  else
    flag=0;            //next char
  p++;                 //shift left
  n=n-m*s;              // reduce divisor
  s/=10;
   }
```

```
p=num1+10-c;


for (i=10-c;i<10;i++) {
  if ((dp!=0) && (i==(10-dp))) lcd_putchar(',');
  lcd_putchar(*p);
  p++;


  }
}


// ÓõíÜñôçóç print_adc.Åßíáé ç óõíÜñôçóç ç ïðïßá ìåôáôñÝðåé ôçí
áíáëïãéêÞ ôÜóç åéóüäïõ óå
// áñéèìü âçìÜôùí ôïõ ADC êáé ìåôÜ óå âáèìïýò Êå ëóßïõ.


void print_adc(){


        char FIFOByte = 0x00;
   char Sync = 0x00;
   unsigned long int temp;
   unsigned int temp1,temp2,temp3,temp4;
   bit recording=1;


   TCNT1=0;    //reset delay timer
   write_data=0;


   while (recording) {
```

```
   putchar(10);putchar(13);


   temp=read_adc(0)-51;        //Êáôá÷ùñïýìå óôçí ìåôáâëçôÞ temp ôçí ôéìÞ
ðïõ Ý÷åé ç ìåôáâëçôÞ

   temp*=10000;                // read_adc áöáéñþíôáò 51 âÞìáôá ðïõ áíôéóôïé÷ïýí
óå 0.25V ðïõ

   temp/=461;                  // åßíáé ç Ýîïäïò ôùí áéóèçôçñßùí óôïõò  -50C ðïõ
åßíáé ç åëÜ÷éóôç

   temp-=5000;                 // ìÝôñçóç ðïõ ìðïñïýìå íá êÜíïõìå.Ï êÜèå âáèìüò
Êåëóßïõ áíôéóôïé÷åß

    clear();                   // óå 4,61  âÞìáôá(922/200).

    lcd_gotoxy(1,0);

     lcd_putsf("T1:");


    if ( ischneg(temp)){          //Óå ðåñßðôùóç ðïõ ç èåñìïêñáóßá åßíáé
ìéêñüôåñç ôïõ ìçäåíüò

      lcd_putsf("-");             // êáëïýìå ôçí ischneg êáé ôõðþíïõìå Ýíá " -" ìðñïóôÜ
áðü ôçí

      temp=(50000-temp); }        // ôéìÞ ðïõ Ý÷åé ç temp êÜíùíôáò ôáõôü÷ñïíá
èåôéêÞ ôçí temp.


   if(temp<=15 && temp>=-15){

    lcd_gotoxy(4,0);

    lcd_putsf("0");

        }


   lcdputnum (temp,4,'0',2);

   lcd_gotoxy(4,0);
```

```
   temp1=(unsigned int) temp;

// Ìîîßùò ìå ôï ðñþôï êáíÜëé ç ßäéá äéáäéêáóßá ãßíåôáé êáé ãéá ôá
// õðüëïéðá ôñßá êáíÜëéá ôïõ ADC.

   temp=read_adc(1)-51;
   temp*=10000;
   temp/=461;
   temp-=5000;

   lcd_gotoxy(11,0);
    lcd_putsf("T2:");

    if ( ischneg(temp)){
      lcd_putsf("-");
      temp=(50000-temp); }


    lcdputnum (temp,4,'0',2);
    lcd_gotoxy(14,0);

   temp2=(unsigned int) temp;

   temp=read_adc(2)-51;
   temp*=10000;
```

```
  temp/=461;
  temp-=5000;

  lcd_gotoxy(1,2);
   lcd_putsf("T3:");


   if ( ischneg(temp)){
     lcd_putsf("-");
     temp=(50000-temp); }


  lcdputnum (temp,4,'0',2);
  lcd_gotoxy(4,2);

  temp3=(unsigned int) temp;

  temp=read_adc(3)-51;
  temp*=10000;
  temp/=461;
  temp-=5000;

 lcd_gotoxy(11,2);
   lcd_putsf("T4:");
```

```
     if ( ischneg(temp)){
       lcd_putsf("-");
       temp=(50000-temp); }



    lcdputnum (temp,4,'0',2);
    lcd_gotoxy(14,2);



  lcd_gotoxy(0,3);
  lcd_putsf("  Press 'A' to Exit");


   temp4=(unsigned int) temp;


// ¸ëåã÷ïò ýðáñîçò óõóêåõÞò USB Flash óõíäåäåìÝíçò óôï VDRIVE.
// H VNC1L_State åßíáé ç ìåôáâëçôÞ ðïõ ìáò êáèïñßæåé ðïßá åßíáé ç êáôÜóôáóç ôïõ VDRIVE.


if (VNC1L_Sync() == Synchronised)          {
                          switch (VNC1L_State)
                          {
  case VNC1L_Idle:
                            Sync = VNC1L_Sync();
 if (VNC1L_FindDisk() == GotDisk)  // ¸ëåã÷ïò ýðáñîçò USB
                                  {
   VNC1L_State = VNC1L_DiskAvailable;   // ÅÜí õðÜñ÷åé USB íá ãßíåé äéáèÝóéìç ç óõóêåõÞ
```

```
                }
                break;

        case VNC1L_DiskAvailable:
    VNC1L_OpenFile();              //ÅÜí VNC1L_State = VNC1L_DiskAvailable
êáëåßôáé ç VNC1L_OpenFile()
                        // ãéá íá äçìéïõñãçèåß ôï áñ÷åßï text óôï USB ìå ôï üíïìá ðïõ
Ý÷ïõìå åðéëÝîåé.

        putchar('W');                   // Send 'W'
        putchar('R');                   // Send 'R'
        putchar('F');                   // Send 'F'
        putchar(' ');                   // Send ' '
        putchar(0x00);                  // Send 0x00 - Number
of bytes to write MSB
        putchar(0x00);                  // Send 0x00
        putchar(0x00);                  // Send 0x00
        putchar(0x0F);                  // Send 0x0E - Number
of  bytes to write LSB
        putchar(0x0D);                  // Send carriage return
    putnum(day,2,'0',0);putchar('-');putnum(month,2,'0',0);putchar('-
');putnum(year,2,'0',0);
    putchar(0x0D);                  // Send carriage return
        putchar(0x0A);                  // Send line feed
    putnum(hours,2,'0',0);putchar(':');putnum(minutes,2,'0',0);
        putchar(0x0D);                  // Send carriage return
        putchar(0x0A);                  // Send line feed
        putchar(0x0D);                  // Send carriage return
```

```
  VNC1L_State = VNC1L_WriteFile; // Update state to indicate file has been
created and can be written

            break;


case VNC1L_WriteFile:          //ÅÜí VNC1L_State =VNC1L_WriteFile ôüôå
áñ÷ßæåé ç äéáäéêáóßá

                        // åããñáöÞò ìÝóá óôï áñ÷åßï ôùí äåäïìÝíùí ðïõ åðéèõìïýìå.


 putchar('W');                                          // Send 'W'

          putchar('R');                                 // Send 'R'

          putchar('F');                                 // Send 'F'

          putchar(' ');                                 // Send ' '

          putchar(0x00);                                        // Send 0x00 - Number
of bytes to write MSB

          putchar(0x00);                                        // Send 0x00

          putchar(0x00);                        // Send 0x00

          putchar(0x19);                                        // Send 0x0E - Number
of  bytes to write LSB

          putchar(0x0D);                                        // Send carriage return

       putnum(temp1,5,' ',2);putnum(temp2,5,' ',2);putnum(temp3,5,'  ',2);

       putnum(temp4,5,' ',2);putchar(10);putchar(13);

          putchar(0x0D);                                        // Send carriage return

          putchar(0x0A);                                        // Send line feed

          putchar(0x0D);                                        // Send carriage return

          while (rx_counter < 0x05)                     //Wait until at least 5 bytes in
the Rx FIFO
```

```
        {
                delay_ms(10);                                   //Wait for 5 byte
response("D:\>" + 0x0D)
        }

        while (rx_counter > 0x00)
        {
                FIFOByte=getchar();                     //Read response bytes form
FIFO
        }

while (!write_data) {
     KeyDecoder();              //Ïðïéáäðïîå óôéãìÞ åðéèõìïý ìå ìðïñïýìå íá
óôáìáôÞóïõìå
  if(key==10 & ButtonPressed==1){   // ôçí áðïèýêåõóç ðáôþíôáò ôï
"A".Ôáõôü÷ñïíá óôáìáôÜåé ç
     ButtonPressed=0;           // äéáäéêáóßá åããñáöÞò óôï text êáé êëåßíåé
ôï text.
     recording=0;
     write_data=1;
      VNC1L_State = VNC1L_EndFile;         // Update state to indicate file has
been written and can now be closed
      VNC1L_CloseFile();
     }
   }
  write_data=0;
     VNC1L_WriteToFile(temp1,temp2,temp3,temp4);   // Send write file
command (WRF) - write "Hello World!"
     //VNC1L_State = VNC1L_EndFile;                  // Update state to
indicate file has been written and can now be closed
```

```
                break;


case VNC1L_EndFile:

    VNC1L_CloseFile();                  // Send close file command (CLF)

    VNC1L_State = VNC1L_WaitForRemove;  // Update state to indicate
witing for disk to be removed

                break;


case VNC1L_WaitForRemove:

    Sync = VNC1L_Sync();

      if (VNC1L_FindDisk() != GotDisk)  // Check for disk

       {

         VNC1L_State = VNC1L_Idle;      // Update state to indicate
disk has been removed and return to idle

              }

               break;


      default:

            VNC1L_State = VNC1L_Idle;   // Default to idle state

                    }


               }


 }


}
```

```
int length(int x){
        int len;

        if(x>=10000)len=5;
   else if(x>=1000 && x<10000) len=4;
   else if(x>=100 && x<1000) len=3;
        else if(x>=10 && x<100) len=2;
        else if(x<10)  len=1;
        return len;
}

// ÓõíÜñôçóç ischneg.×ñçóéìïðïéåßôáé óôçí ðåñßðôùóç ðïõ ç ôéìÞ
// ôïõ ADC ãßíåôáé áñíçôéêÞ.

int ischneg(int temp){
  char ischneg ;
  if(temp<50){
   ischneg=1;
  }
  else {
   ischneg=0;
  }

  return ischneg;
}
```

```
//*****************************************************************************
***
//
//  Define states for state machine
//
//*****************************************************************************
***
enum {                                  // Enum states
        VNC1L_Idle,                 // Idle state
        VNC1L_DiskAvailable,
        VNC1L_CreateFile,
        VNC1L_WriteFile,
        VNC1L_EndFile,
        VNC1L_WaitForRemove
};

char VNC1L_State = VNC1L_Idle;          // Initialise state to idle

char Synchronised = 0x01;
char GotDisk = 0x01;

//*****************************************************************************
***
//
//  Echo command using E - same for short command set or long command
set
//
```

```
//*****************************************************************************
***

void VNC1L_Big_E()

{

        putchar('E');                   // Send 'E'

        putchar(0x0D);                      // Send carriage return

}




//*********************************************************************************
***

//

//  Echo command using e - same for short command set or long command
set

//

//*****************************************************************************
***

void VNC1L_Small_e()

{

        putchar('e');                   // Send 'e'

        putchar(0x0D);                      // Send carriage return

}
```

```
//*****************************************************************************
***
//
//  Synchronise to the VNC1L using the e and E echo commands
//
//*****************************************************************************
***
char VNC1L_Sync()
{
        char LoopCount = 0x00;
        char FIFOByte1 = 0x00;
        char FIFOByte2 = 0x00;
        char Got_Big_E = 0x00;
        char Got_Small_e = 0x00;

//      serial_resetrxfifo();           // Reset FIFO before synchronising
 rx_wr_index=rx_rd_index=rx_counter=0;

        while ((Got_Big_E == 0x00) && (LoopCount < 0xFF))
        {
                VNC1L_Big_E();                  // Send Big E echo
                LoopCount++;                // Increment loop count
                while (rx_counter < 0x02)
                {
                        delay_ms(100);      // Wait for 2 byte response
                }
```

```
                if (rx_counter > 0x00)          // If data is available
                {
      while (rx_counter > 0x02)  // Read all data available except last 2 bytes
                 {
                FIFOByte1=getchar();          // If more than 2 bytes available,
read surplus ones


                }
                FIFOByte1=getchar();   // Check that remaining 2 bytes are 'E'
and 0x0D
                FIFOByte2=getchar();
      if ((FIFOByte1 == 'E') && (FIFOByte2 == 0x0D))
                {
                Got_Big_E = 0x01;
                 }
                else
                {
      delay_ms(10);              // Wait a bit and retry synchronisation
                        }
                }
        }


        if (Got_Big_E == 0x01)
        {
                VNC1L_Small_e();
                while (rx_counter < 0x02)
                {
```

```
                delay_ms(10);           // Wait for 2 byte response
            }
            FIFOByte1=getchar();        // Check that remaining 2 bytes are
'e' and 0x0D
            FIFOByte2=getchar();
            if ((FIFOByte1 == 'e') && (FIFOByte2 == 0x0D))
            {
        Got_Small_e = 0x01;  // If small e found, then synchronised
            }

        }

        return Got_Small_e;

}

//*****************************************************************************
***
//
//  Look for disk - assumes long command set being used
//
//*****************************************************************************
***
char VNC1L_FindDisk()               // Return 0x01 if disk is found, else return
0x00
{
        char FIFOByte1 = 0x00;
        char FIFOByte2 = 0x00;
```

```
        char FIFOByte3 = 0x00;
        char FIFOByte4 = 0x00;
        char FIFOByte5 = 0x00;

        putchar(0x0D);                  // Send carriage return
        while (rx_counter < 0x05); // Wait until at least 5 bytes in the Rx FIFO
        FIFOByte1=getchar();          // Read bytes out of Rx FIFO
        FIFOByte2=getchar();   // Read bytes out of Rx FIFO
        FIFOByte3=getchar();  // Read bytes out of Rx FIFO
        FIFOByte4=getchar();  // Read bytes out of Rx FIFO
        FIFOByte5=getchar(); // Read bytes out of Rx FIFO

// Check for prompt
if ((FIFOByte1 == 'D') && (FIFOByte2 == ':') && (FIFOByte3 == 0x5C) &&
(FIFOByte4 == '>') && (FIFOByte5 == 0x0D))
        {
                return 0x01;    // If prompt found, then return disk available
        }
      else
        {
while (rx_counter > 0x00)
   {
  FIFOByte1=getchar();          // Read any additional bytes out of the FIFO
                }
                return 0x00;    // Return no disk available
        }
```

```
}

//*****************************************************************************
***
//
//  Open file for write command (OPW) using long command set
//
//*****************************************************************************
***
void VNC1L_OpenFile()
{
        char FIFOByte = 0x00;

        putsf("OPW Times.txt");putchar(10);putchar(13);
/*
        putchar('O');                   // Send 'O'
        putchar('P');                   // Send 'P'
        putchar('W');                   // Send 'W'
        putchar(' ');                   // Send ' '
        putchar('M');                   // Send 'M'
        putchar('e');                   // Send 'e'
        putchar('a');                   // Send 'a'
        putchar('s');                   // Send 's'
```

```
        putchar('u');                   // Send 'u'
        putchar('r');                   // Send 'r'
        putchar('m');                   // Send 'm'
        putchar('e');                   // Send 'e'
        putchar('n');                   // Send 'n'
        putchar('t');                   // Send 't'
        putchar('s');                   // Send 's'
        putchar('.');                   // Send '.'
        putchar('t');                   // Send 't'
        putchar('x');                   // Send 'x'
        putchar('t');                   // Send 't'
        putchar(0x0D);                          // Send carriage return
    */
        while (rx_counter < 0x05) // Wait until at least 5 bytes in the Rx FIFO
        {
                delay_ms(10);  // Wait for 5 byte response("D: \>" + 0x0D)
        }
        while (rx_counter > 0x00)
        {
                FIFOByte=getchar(); // Read response bytes form FIFO
        }
}
```

```
//*****************************************************************************
***
//
//  Write to file command (OPW) using long command set
//
//*****************************************************************************
***
void VNC1L_WriteToFile(unsigned int temp1, unsigned int temp2, unsigned int
temp3, unsigned int temp4, )
{
        char FIFOByte = 0x00;


//      putchar('W');       // Send 'W'
//      putchar('R');       // Send 'R'
//      putchar('F');       // Send 'F'
//      putchar(' ');       // Send ' '
//
//
//
//      putchar(0x00);              // Send 0x00 - Number of bytes to write MSB
//      putchar(0x00);              // Send 0x00
//      putchar(0x00);              // Send 0x00
//      putchar(0x0E);              // Send 0x0E - Number of  bytes to write LSB
//      putchar(0x0D);              // Send carriage return
//
// //   printf (" %d %d %d %d",temp1,temp2,temp3,temp4);
// //    putchar(10);
```

```
// //    putchar(13);
// //    //-------------------------------------------------
// //    /*
// //    putchar('H');          // Send 'H'
// //    putchar('e');        // Send 'e'
// //    putchar('l');          // Send 'l'
// //    putchar('l');        // Send 'l'
// //    putchar('o');          // Send 'o'
// //    putchar(' ');          // Send ' '
// //    putchar('W');          // Send 'W'
// //    putchar('o');          // Send 'o'
// //    putchar('r');          // Send 'r'
// //    putchar('l');          // Send 'l'
// //    putchar('d');          // Send 'd'
// //    putchar('!');          // Send '1'
// //  */
// //  //-------------------------------------------------
// //
//        putchar(0x0D);          // Send carriage return
//        putchar(0x0A);          // Send line feed
//        putchar(0x0D);          // Send carriage return
//
        while (rx_counter < 0x05)    // Wait until at least 5 bytes in the Rx FIFO
        {
                delay_ms(10);         // Wait for 5 byte response("D:\>" + 0x0D)
        }
```

```
        while (rx_counter > 0x00)
        {
                FIFOByte=getchar(); // Read response bytes form FIFO
        }

}

//*****************************************************************************
***
//
//  Close file command (CLF) using long command set
//
//*****************************************************************************
***
void VNC1L_CloseFile()
{
        char FIFOByte = 0x00;

        putsf("CLF Times.txt");putchar(13);
/*
        putchar('C');                   // Send 'C'
        putchar('L');                   // Send 'L'
```

```
        putchar('F');                   // Send 'F'
        putchar(' ');                   // Send ' '
        putchar('h');                   // Send 'h'
        putchar('e');                   // Send 'e'
        putchar('l');                   // Send 'l'
        putchar('l');                   // Send 'l'
        putchar('o');                   // Send 'o'
        putchar('.');                   // Send '.'
        putchar('t');                   // Send 't'
        putchar('x');                 // Send 'x'
        putchar('t');                   // Send 't'
        putchar(0x0D);                          // Send carriage return
 */
        while (rx_counter < 0x05)    // Wait until at least 5 bytes in the Rx FIFO
        {
                delay_ms(10);                   // Wait for 5 byte response("D:\>" +
0x0D)
        }

        while (rx_counter > 0x00)
        {
                FIFOByte=getchar(); // Read response bytes form FIFO
        }

}
```