Ο Arduino στο Εργαστήριο Φυσικών Επιστημών

Β. Νούσης[1], Β. Νούση[2]

[1]Υπεύθυνος ΕΚΦΕ Θεσπρωτίας
bill1961gr@yahoo.gr
[2]Φοιτήτρια Πληροφορικής ΑΠΘ
vivinousi@yahoo.gr

Περίληψη

Στην παρούσα εργασία διερευνώνται οι δυνατότητες της πλατφόρμας Arduino στον τομέα της συλλογής δεδομένων μέσω αναλογικών και ψηφιακών αισθητήρων. Δίνεται έμφαση σε προγραμματιστικές τεχνικές με σκοπό τη βελτίωση της ποιότητας των μετρήσεων, την αύξηση του ρυθμού δειγματοληψίας, καθώς και του ρυθμού μετάδοσης των δεδομένων στο σύστημα απεικόνισης. Παρουσιάζεται εν τέλει το πρωτότυπο ενός συστήματος συγχρονικής λήψης και απεικόνισης (ΣΣΛΑ) βασισμένο στην πλατφόρμα Arduino.

Λέξεις κλειδιά: Arduino, αισθητήρας, μετατροπέας αναλογικού σε ψηφιακό, ρυθμός δειγματοληψίας, σύστημα συγχρονικής λήψης και απεικόνισης.

1. Εισαγωγή

Η πλατφόρμα Arduino είναι ένα ανοιχτού υλικού και ανοιχτού κώδικα σύστημα ανάπτυξης ηλεκτρονικών πρωτοτύπων, βασισμένο σε ευέλικτο και εύκολο στη χρήση υλικό και λογισμικό.

Το μοντέλο Arduino Uno υλοποιείται γύρω από τον οκτάμπιτο μικροελεγκτή AT-mega328P της Atmel χρονισμένο στα 16MHz, ο οποίος διαθέτει: 14 ψηφιακές εισόδους-εξόδους, 6 πολυπλεγμένες αναλογικές εισόδους διακριτικής ικανότητας 10bit, μνήμη 32 kB τύπου Flash, 1 kB EEPROM και 2 kB SRAM. Στο υλικό του μικροελεγκτή συμπεριλαμβάνονται τρεις «χρονιστές»: timer0 και timer2 διακριτικής ικανότητας 8 bit και ο timer1 των 16 bit (Atmel 2013). Χρησιμεύουν είτε στην εισαγωγή καθυστέρησης κατά την εκτέλεση των εντολών του προγράμματος είτε στην εκτέλεση συγκεκριμένων εντολών σε τακτά χρονικά διαστήματα

Μερικές από τις ψηφιακές εισόδους/εξόδους, που καταλήγουν σε αντίστοιχες ακίδες στην πλακέτα του Arduino, έχουν και ειδικές χρήσεις, π.χ: οι ακίδες 3, 5, 6, 9, 10, 11 λειτουργούν ως «ψευδοαναλογικές» έξοδοι (PWM) των 8-bit, ενώ οι ακίδες 2 και 3 μπορούν να ρυθμιστούν ώστε να προκαλέσουν εξωτερική διακοπή (interrupt) του εκτελούμενου κώδικα στον μικροελεγκτή. Αντίστοιχα οι έξι αναλογικές είσοδοι μπορούν να χρησιμοποιηθούν ως ψηφιακές είσοδοι/έξοδοι (ψηφιακές ακίδες 14 μέχρι

19), και οι αναλογικές είσοδοι A4 και A5 με χρήση του κατάλληλου λογισμικού χρησιμοποιούνται για την υλοποίηση του δισύρματου πρωτοκόλλου επικοινωνίας I^2C. Στην πλακέτα του Arduino, ένας δεύτερος μικροελεγκτής (Atmega 16U2) κατάλληλα προγραμματισμένος λειτουργεί ως μετατροπέας σειριακού σε USB για την επικοινωνία του Arduino με ηλεκτρονικό υπολογιστή (Margolis Michael, 2012).

Από πλευράς λογισμικού, η πλατφόρμα συνοδεύεται από ένα αριθμό βιβλιοθηκών πυρήνα που παρέχουν επιπλέον λειτουργικότητα στη διαχείριση του υλικού και των δεδομένων και βελτιώνουν την αναγνωσιμότητα του κώδικα, καθώς και από το ολοκληρωμένο περιβάλλον εργασίας (Arduino IDE) μια εφαρμογή σε Java που περιλαμβάνει ένα επεξεργαστή κειμένου για τη συγγραφή των προγραμμάτων, εκτελεί τη μεταγλώττιση του προγράμματος σε κώδικα κατανοητό από τον μικροελεγκτή, ενώ έχει και τη δυνατότητα να «φορτώνει» τον κώδικα στον Arduino. Περιλαμβάνει επίσης μια σειριακή κονσόλα μέσω της οποίας ο χρήστης μπορεί να αλληλεπιδρά με το σύστημα Arduino στέλνοντας κατάλληλες εντολές ή λαμβάνοντας δεδομένα

2. Σύστημα συγχρονικής λήψης και απεικόνισης

Σε γενικές γραμμές ένα σύστημα συγχρονικής λήψης και απεικόνισης αποτελείται: από ένα σύνολο αισθητήρων με τα κατάλληλα κυκλώματα προσαρμογής, ένα σύστημα συλλογής δεδομένων (ο Arduino στην περίπτωσή μας) και ένα σύστημα απεικόνισής τους (υπολογιστής και κατάλληλο λογισμικό).

2.1 Αναλογικοί αισθητήρες – Προσαρμογή σήματος

Πρόκειται για ηλεκτρονικές ή ηλεκτρομηχανικές διατάξεις στις οποίες μια κατάλληλη ηλεκτρική ποσότητα (τάση, ρεύμα, αντίσταση, χωρητικότητα) μεταβάλλεται ανάλογα με την τιμή ενός φυσικού μεγέθους. Ενδεικτικά και μόνο, αναφέρουμε τους αισθητήρες: τάσης, θερμοκρασίας, πίεσης, δύναμης, επιτάχυνσης, pH, κ.ά.

Η σύνδεση ενός αισθητήρα σε μια από τις αναλογικές εισόδους του Arduino απαιτεί ένα κύκλωμα προσαρμογής (Putnam και Knapp, 1996), που ο ρόλος του γενικά συνοψίζεται στα εξής:

- Μετατροπή του προς μέτρηση φυσικού μεγέθους σε αναλογική τάση. Ιδανικά η τάση αυτή πρέπει να μεταβάλλεται γραμμικά με την τιμή του μετρούμενου μεγέθους στην περιοχή 0 - V_{ref}, όπου V_{ref} τάση αναφοράς του χρησιμοποιούμενου για τη συλλογή των δεδομένων αναλογικο-ψηφιακού μετατροπέα (ADC). Η τάση αναφοράς στον Arduino έχει ρυθμιστεί στην τιμή V_{ref} =5V που σημαίνει διακριτική ικανότητα 5/1024 V ≃ 4.9 mV ανά μονάδα. Η εντολή analogReference() της γλώσσας προγραμματισμού του Arduino, ρυθμίζει την τάση αναφοράς εσωτερικά στα 1,1 V ή σε

οποιαδήποτε άλλη τιμή (μεταξύ 0-5V) εξωτερικής τάσης που θα συνδεθεί στην ακίδα AREF, αυξάνοντας έτσι τη διακριτική ικανότητα του συστήματος αλλά μειώνοντας ταυτόχρονα το εύρος της μετρούμενης τάσης (Arduino, 2013).

- Προσαρμογή της αντίστασης εξόδου του αισθητήρα σε τιμή μικρότερη των 10 kΩ ώστε να είναι πολύ μικρός ο χρόνος φόρτισης του πυκνωτή Sample/Hold που υπάρχει στην είσοδο του ADC. Μεγαλύτερες αντιστάσεις δημιουργούν προβλήματα στην ακρίβεια της A/D μετατροπής (Atmel, 2013).
- Απομάκρυνση ή μείωση των επιπέδων των ανεπιθύμητων σημάτων αλλά και μείωση του φασματικού εύρους του προς μετατροπή αναλογικού σήματος στο μισό τουλάχιστον της συχνότητας δειγματοληψίας (Θεώρημα Nyquist).

2.2 Συλλογή δεδομένων

2.2.1 Ρυθμός δειγματοληψίας

Η συλλογή των δεδομένων γίνεται με δειγματοληψία στις αναλογικές εισόδους του Arduino. Με την εντολή analogRead() μπορούμε να «διαβάσουμε» το αποτέλεσμα μιας A/D μετατροπής από μια συγκεκριμένη αναλογική είσοδο. Καθώς ένα ΣΣΛΑ πρέπει να συλλέγει δεδομένα σε τακτά χρονικά διαστήματα προκύπτει η ανάγκη να διερευνήσουμε πόσο μικρά μπορεί να είναι αυτά τα χρονικά διαστήματα ή αλλιώς ποιος μπορεί να είναι ο μέγιστος ρυθμός δειγματοληψίας. Η συχνότητα χρονισμού του ADC στον Arduino ρυθμίζεται μέσω κατάλληλου διαιρέτη (prescaler) σε τιμή υποπολλαπλάσιο της συχνότητας χρονισμού του Arduino. Εξ' ορισμού ο διαιρέτης έχει την τιμή 128 που χρονίζει το ADC στα (16/128) MHz ή 125 kHz ενώ σύμφωνα με τον κατασκευαστή (Atmel, 2013) κάθε μετατροπή διαρκεί 13 κύκλους, δηλ. 104μs, που σημαίνει ένα μέγιστο ρυθμό δειγματοληψίας 9,6kHz. Προφανώς μπορούμε να αυξήσουμε το ρυθμό δειγματοληψίας μειώνοντας την τιμή του διαιρέτη του ADC.

Το ακόλουθο πρόγραμμα, χρησιμοποιώντας μόνο τις standard εντολές της γλώσσας προγραμματισμού του Arduino (Arduino, 2013), δίνει μια πρώτη ιδέα των δυνατοτήτων της πλατφόρμας στον τομέα του ρυθμού δειγματοληψίας:

```
//ADC speed test
void setADCPrescaler(uint8_t prescaler) {
        const uint8_t ADPS_MASK = ~0x07;
        ADCSRA = (ADCSRA & ADPS_MASK) | prescaler;
}
void setup() {
        Serial.begin(115200);           //Ενεργοποίηση σειριακής επικοινωνίας
        setADCPrescaler(7);             //Χρονισμός ADC στα 125kHz
}
```

```
void loop() {
        long start = micros();                  //Αρχή χρόνου
        for (int i = 0; i < 1000; i++) {        //1000 αναλογικές μετατροπές
            long j = analogRead(0);
        }
        long time = micros() - start;       //Χρονική διάρκεια 1000 μετατροπών
        Serial.print("Time (us): ");        //Σειριακή αποστολή αποτελεσμάτων
        Serial.print(time);
        while (true);                       //Αναμονή
}
```

Ουσιαστικά μετράμε το χρόνο που απαιτείται για 1000 διαδοχικές A/D μετατροπές στην αναλογική είσοδο 0. Η συνάρτηση setADCPrescler() ανάλογα με την τιμή (0, …, 7) της μεταβλητής prescaler καθορίζει την τιμή του διαιρέτη σε 4, 8, 16, .., 128 αντίστοιχα. Οι τιμές που πετύχαμε όσον αφορά το ρυθμό δειγματοληψίας, φαίνονται στον πίνακα 1 που ακολουθεί:

Πίνακας 1: *Μέγιστος ρυθμός δειγματοληψίας*

Prescaler	Συχνότητα ADC (kHz)	Ρυθμός δειγματοληψίας (kHz)
128	125	8,9
64	250	17,8
32	500	33,2
16	1000	58,5
8	2000	99,6
4	4000	153,2

Πρέπει να τονιστεί πως τα αποτελέσματα αυτά για το μέγιστο ρυθμό δειγματοληψίας μόνο θεωρητική αξία έχουν αφού:

- Η ποιότητα της A/D μετατροπής εξαρτάται από τη συχνότητα χρονισμού του ADC. Σύμφωνα με τον κατασκευαστή για καλύτερα αποτελέσματα η συχνότητα χρονισμού του ADC δεν πρέπει να υπερβαίνει τα 200 kHz, αν και για συχνότητες μέχρι και 1 MHz η ανάλυση δε μειώνεται σημαντικά
- Δεν έχει συνυπολογιστεί ο χρόνος που απαιτείται για την αποθήκευση των μετρήσεων, καθώς η ταχύτατη αλλά περιορισμένου μεγέθους RAM του Arduino (2 kb) δεν επιτρέπει τη χρήση του για συνεχή δειγματοληψία.

Για τη επίτευξη ενός σταθερού ρυθμού δειγματοληψίας η απλούστερη λύση είναι η χρήση μιας χρονοκαθυστέρησης μεταξύ των διαδοχικών μετατροπών με χρήση της εντολής delay() ή της delayMicroseconds(). Καθώς όμως κατά τη διάρκεια της καθυστέρησης καμιά άλλη λειτουργία δε μπορεί να εκτελέσει ο μικροελεγκτής (Arduino, 2013) η επιλογή αυτή κρίνεται ακατάλληλη. Η λύση που προτιμείται είναι η

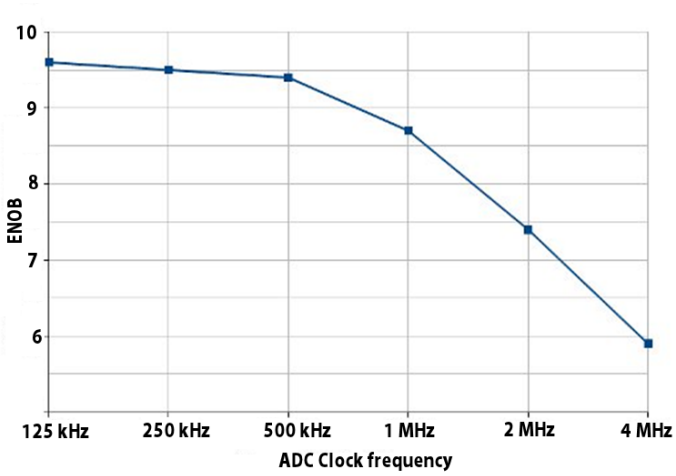
χρήση διακοπών χρονιστή (timer interrupts). Η ελεύθερα διανεμόμενη βιβλιοθήκη FlexiTimer2 υλοποιεί με απλό τρόπο αυτή τη δυνατότητα χρησιμοποιώντας τον Timer2, όπως φαίνεται και στο πρόγραμμα που ακολουθεί, με το οποίο δειγματοληπτούμε στην αναλογική είσοδο A0 με σταθερό ρυθμό 100 Hz ή μία μέτρηση ανά 10 ms:

```
//Example Sketch for SampleRate = 100 Hz
#include <FlexiTimer2.h>
void runADC() {
        int value0 = analogRead(0);              //Do something with value0….
}
void setup() {
          Serial.begin(500000);
          FlexiTimer2::set(10, runADC);          //Ρυθμός δειγματοληψίας = 100 Hz
}
void loop() {
          FlexiTimer2::start();                  //Ενεργοποίηση του Timer
          while (Serial.available() == 0);       //Αναμονή
          FlexiTimer2::stop();                   //Απενεργοποίηση του Timer
}
```

Είναι ιδιαίτερα σημαντικό να ληφθεί μέριμνα ώστε η συνάρτηση που εκτελείται όταν ενεργοποιηθεί η διακοπή χρονιστή να είναι όσο το δυνατό λιγότερο χρονοβόρα.

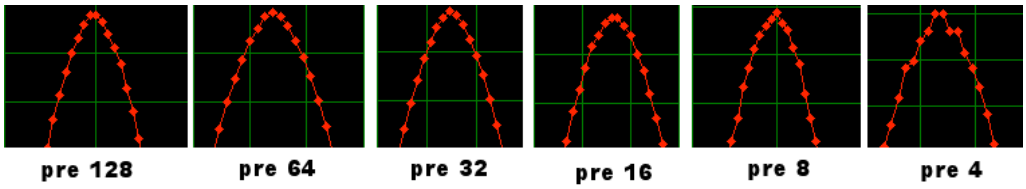
2.2.2. Ποιότητα μετατροπής

Ένας από τους τρόπους υπολογισμού της ποιότητας μιας A/D μετατροπής είναι το μέγεθος ENOB (Effective Number Of Bits – ενεργός αριθμός των bits) που ορίζεται ως: $ENOB = \frac{SINAD\ 2\,1{,}76}{6{,}02}$, όπου SINAD είναι ο λόγος του σήματος προς το θόρυβο και την παραμόρφωση του μετατροπέα. Στην περίπτωση του Arduino όσο πιο κοντά στο 10 βρεθεί η τιμή του ENOB τόσο καλύτερη η ποιότητα της μετατροπής. Έχουμε ήδη αναφέρει πως η ποιότητα της A/D μετατροπής ε-ξαρτάται από τη συχνότη-τα χρονισμού του ADC. Η διπλανή γραφική παρά-σταση (Εικόνα 1) αναφέ-ρεται σε αυτήν ακριβώς τη σχέση



Εικόνα 1: *ENOB - ADC Clock frequency*

(OpenmusicLabs, 2013). Είναι φανερό πως για συχνότητες χρονισμού του ADC μέχρι 500 kHz (prescaler 32) η μείωση στην ποιότητα της μετατροπής είναι πολύ μικρή, ενώ οριακά μπορεί να θεωρηθεί ικανοποιητική ακόμη και μέχρι το 1 MHz. Στην εικόνα 2 φαίνεται η μεταβολή στην ποιότητα της ψηφιοποίησης ενός ημιτονικού σήματος για τις διαφορετικές τιμές του διαιρέτη του ADC:



Εικόνα 2: *Ψηφιοποιημένο σήμα για διάφορες τιμές της συχνότητας του ADC*

Μικρή βελτίωση στην ποιότητα της μετατροπής προκύπτει με την απενεργοποίηση της δυνατότητας να χρησιμοποιηθούν οι αναλογικές είσοδοι και ως ψηφιακές, αφού μια ελεύθερη (μη συνδεδεμένη) ψηφιακή είσοδος μπορεί με τυχαίο τρόπο να αλλάζει κατάσταση (από LOW σε HIGH ή αντίστροφα) δημιουργώντας επιπλέον ψηφιακό θόρυβο στην περιοχή της αναλογικής εισόδου. Επιπλέον η μικρή χωρητικότητα της ψηφιακής εισόδου μπορεί προκαλέσει σφάλματα μετατροπής (OpenmusicLabs, 2013). Η απενεργοποίηση αυτής της δυνατότητας στις αναλογικές εισόδους A0, …, A3 γίνεται μέσω του καταχωρητή DIDR0 του μικροελεγκτή, ως εξής: DIDR0 = DIDR0 | B00001111.

Στην περίπτωση σημάτων DC ή αργά μεταβαλλόμενων AC (π.χ. θερμοκρασία) οι επιπτώσεις του τυχαίου θορύβου στο σήμα μπορούν να ελαχιστοποιηθούν με τη χρήση ψηφιακών φίλτρων (εξομάλυνση μέσου όρου, κινούμενου μέσου όρου, διάμεσου). Στην απλούστερη αλλά και λιγότερο αποδοτική υλοποίηση ένα τέτοιο φίλτρο συνίσταται στη λήψη ενός μεγάλου αριθμού μετρήσεων (συνήθως πολλαπλάσιο του 2) και εύρεση του μέσου όρου τους (Wheat, 2011), π.χ.:

```
int averagingADC() {
        int value0 = 0;
        for (int i=0; i<16; i++) {
          value0 += analogRead(0);
        }
        return (value0 >> 4);      //Ιδιο με το: value0 / 16
}
```

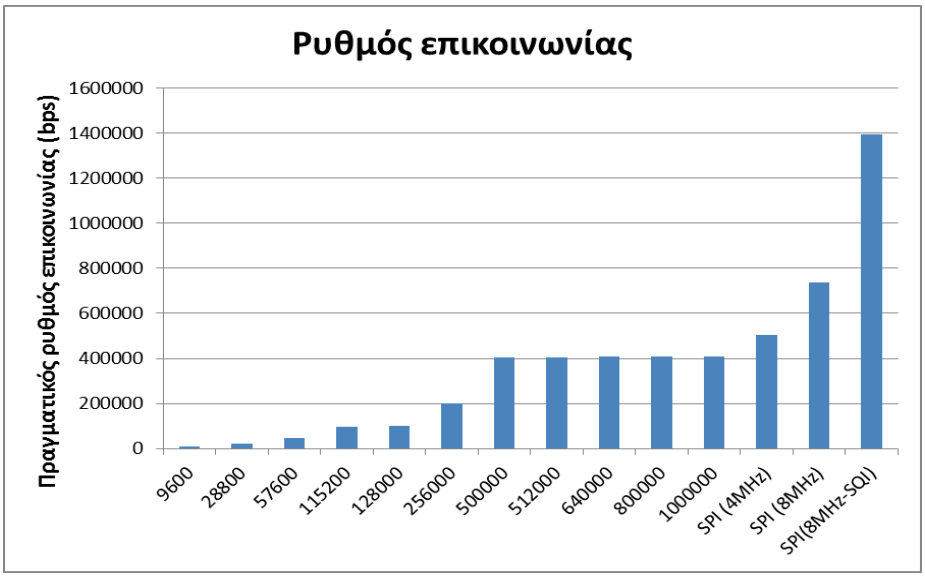
Στην εξομάλυνση κινούμενου μέσου όρου η τιμή της μέτρησης προκύπτει από το μέσο όρο ενός αριθμού (π.χ. 3) προηγουμένων μετρήσεων και της κάθε φορά νέας μέτρησης, ενώ στην εξομάλυνση διαμέσου γίνεται αύξησα ταξινόμηση ενός αριθμού μετρήσεων και υπολογίζεται ο μέσος όρος της μέγιστης και ελάχιστης μέτρησης.

Τέλος με την τεχνική της υπερδειγματοληψίας αξιοποιείται η ύπαρξη θορύβου στο ψηφιοποιούμενο σήμα για να αυξήσουμε την ανάλυση του ADC και συνεπώς και το λόγο του σήματος προς το θόρυβο και την παραμόρφωση. Για παράδειγμα το άθροισμα 4^N διαδοχικών μετρήσεων των 10 bits διαιρείται δια 2^N (δεξιά ολίσθηση κατά N) ώστε να ληφθεί μια τιμή των (10 + N) bits, αυξάνοντας έτσι την ανάλυση του ADC κατά N bits (AVR121, 2005).

2.2.3. Αποστολή δεδομένων στον υπολογιστή

Είναι φανερό πως μόνο ένας πολύ μικρός αριθμός μετρήσεων μπορεί να αποθηκευτεί στη RAM του Arduino, λόγω του περιορισμένου μεγέθους της. Οι λύσεις που έχουμε είναι: είτε η χρήση εξωτερικής μνήμης, για παράδειγμα SD Card ή μνήμη SPI, για αποθήκευση και εν συνεχεία αποστολή των μετρήσεων στο σύστημα απεικόνισης, είτε η απευθείας σειριακή αποστολή των μετρήσεων στο σύστημα απεικόνισης. Η δεύτερη λύση -η προς το παρόν επιλογή μας- έχει νόημα εφόσον η μεταφορά μπορεί να γίνει αρκετά γρήγορα ώστε να μην έχουμε απώλεια δεδομένων λόγω του υψηλού ρυθμού δειγματοληψίας. Η χρήση διακοπών από το λογισμικό του Arduino για τη σειριακή επικοινωνία είναι καθοριστικής σημασίας για το σκοπό μας, ενώ καθώς η σειριακή επικοινωνία Arduino-PC υλοποιείται μέσω ενός μετατροπέα σειριακού σε USB μπορούμε να χρησιμοποιήσουμε και μη προκαθορισμένους ρυθμούς επικοινωνίας εκμεταλλευόμενοι την υψηλή ταχύτητα μεταφοράς του διαύλου USB.

Διερευνώντας τις δυνατότητες του Arduino μετρήσαμε το χρόνο που απαιτείται για τη σειριακή αποστολή 10.000 χαρακτήρων στον μέσω USB συνδεδεμένο υπολογιστή για διάφορες τιμές του ρυθμού επικοινωνίας. Στο διάγραμμα που ακολουθεί φαίνονται τα αποτελέσματα. Οι τρεις τελευταίες μάρες του ιστογράμματος αφορούν τις δοκιμές μας με μνήμη SPI (μοντέλο 23LC1024 της Microchip) σε διάφορους τρόπους λειτουργίας:



***Εικόνα 3**: Πραγματικός ρυθμός αποστολής δεδομένων*

Όπως παρατηρούμε ο πραγματικός ρυθμός επικοινωνίας πρακτικά σταθεροποιείται σε μια τιμή λίγο πάνω από τα 400.000 bps όταν μέσω της εντολής Serial.begin() καθορίζουμε ρυθμούς επικοινωνίας μεγαλύτερους από 500.000 bps, ενώ ο ρυθμός αποστολής δεδομένων στη μνήμη SPI είναι μέχρι και 3,5 φορές μεγαλύτερος. Βέβαια σημαντικό μειονέκτημα της μνήμης SPI είναι το περιορισμένο μέγεθός της (128 kB στην περίπτωση του 23LC1024 που αρκεί για περίπου 6,5 s συνεχούς δειγματοληψίας στα 10 kHz), ενώ θα πρέπει εν συνεχεία τα δεδομένα να μεταφερθούν σειριακά στον υπολογιστή.

Εφόσον το υλικό του Arduino θέτει περιορισμούς στην ταχύτητα μετάδοσης, μπορούμε να επιταχύνουμε τη διαδικασία αν για κάθε μέτρηση στέλνουμε μικρότερο αριθμό χαρακτήρων, χωρίς εννοείται απώλεια πληροφορίας. Ο κώδικας που ακολουθεί δείχνει δυο πιθανούς τρόπους αποστολής των δεδομένων στον υπολογιστή:

1ος τρόπος	**2ος τρόπος**

void setup() {
  Serial.begin(500000);
}
 	
void loop() {
  word value0 = analogRead(0);
  Serial.print(value0);
  Serial.print('s');
 	
	void setup() {
	  Serial.begin(500000);
	}
	
	void loop() {
	  word value0 = analogRead(0);
	  Serial.write(lowByte(value0));
	  Serial.write(highByte(value0));

delay(100); }	delay(100); }

Στην πρώτη περίπτωση η μεγέθους 2 bytes τιμή που «διαβάσαμε» από την αναλογική είσοδο A0 αποστέλλεται στον υπολογιστή μέσω της εντολής Serial.print() ως κείμενο ASCII ακολουθούμενο από τον χαρακτήρα 's' για να σηματοδοτήσει το τέλος της αποστολής. Αυτό κρίνεται απαραίτητο αφού όταν λαμβάνουμε τα δεδομένα στον υπολογιστή δεν είμαστε σε θέση γνωρίζουμε εκ των προτέρων το πλήθος των χαρακτήρων που αναμένουμε. Στην πλειονότητα των περιπτώσεων αυτός ο τρόπος σημαίνει τη σειριακή αποστολή τεσσάρων χαρακτήρων (τρεις για τη μέτρηση αυτή καθ' εαυτή και ένας ο χαρακτήρας τέλους). Αντίθετα στη δεύτερη περίπτωση απαιτούνται μόνο δύο bytes ανά μέτρηση, γεγονός που οδηγεί σε αποστολή διπλάσιου αριθμού δεδομένων στον ίδιο χρόνο. Ας σημειωθεί πως και στις δύο περιπτώσεις οι τιμές που αποστέλλονται στον υπολογιστή είναι αυτές ακριβώς που έχουμε πάρει από τον μετατροπέα A/D χωρίς καμιά επεξεργασία. Επαφίεται στο σύστημα απεικόνισης, η μετατροπή τους σε τιμές που αφορούν το κάθε φορά μετρούμενο φυσικό μέγεθος.

3. Σύστημα απεικόνισης

Ο απλούστερος τρόπος για τη λήψη των δεδομένων στο PC είναι η χρήση του σειριακού τερματικού που περιλαμβάνεται στο IDE του Arduino, που όμως υποστηρίζει μόνο τους προκαθορισμένους ρυθμούς σειριακής επικοινωνίας μέχρι τα 115.200 bps γεγονός που όπως έχουμε ήδη πει έχει σημαντική επίπτωση στο ρυθμό δειγματοληψίας και δε διαθέτει δυνατότητα αποθήκευσης των λαμβανομένων δεδομένων για παραπέρα επεξεργασία. Μπορούμε βέβαια να χρησιμοποιήσουμε κάποιο εξωτερικό τερματικό, π.χ. το Termite είναι μια απλή λύση χωρίς τα ανωτέρω μειονεκτήματα. Τα αποθηκευμένα σε αρχείο κειμένου δεδομένα μπορούν να διαβαστούν από εφαρμογές όπως το EXCEL ή το MATLAB για γραφική αναπαράσταση ή όποια άλλη επεξεργασία κρίνεται απαραίτητη. Γενικά όμως η χρήση κάποιου τερματικού παρουσιάζει κάποια σημαντικά μειονεκτήματα, αφού ο έλεγχος του Arduino πρέπει να γίνεται χειροκίνητα με αποστολή των κάθε φορά κατάλληλων χαρακτήρων, ενώ για τη γραφική αναπαράσταση των δεδομένων απαιτείται εξωτερική εφαρμογή π.χ. EXCEL.

Τόσο για το EXCEL όσο και για το MATLAB υπάρχουν εξειδικευμένα πρόσθετα με κάποιες δυνατότητες ελέγχου του Arduino αλλά και δυνατότητες λήψης και απεικόνισης των δεδομένων. Δεν είχαμε στη διάθεσή μας το πρόσθετο για το MATLAB για δοκιμή, αλλά είχαμε το PLX_DAQ, μια εφαρμογή σε Visual Basic για το EXCEL που διατίθεται ελεύθερα από την Parallax. Σημαντικό μειονέκτημα της εφαρμογής, η υποστήριξη μόνο προκαθορισμένων ρυθμών σειριακής επικοινωνίας μέχρι 128.000 bps με αποτέλεσμα τον περιορισμό του ρυθμού δειγματοληψίας.

Η περιορισμένη εκμετάλλευση των δυνατοτήτων του Arduino από τις έτοιμες εφαρμογές που δοκιμάσαμε μας οδήγησε στον προγραμματισμό σε περιβάλλον Delphi της εφαρμογής datArdu. Η ανάπτυξη της εφαρμογής στηρίχθηκε στην ελεύθερα διατιθέμενη βιβλιοθήκη ComPort που υποστηρίζει σειριακή επικοινωνία με οποιονδήποτε ρυθμό επικοινωνίας και στην ελεύθερη για προσωπική χρήση έκδοση της βιβλιοθήκης PlotLab για τη γραφική αναπαράσταση των δεδομένων.

Στις δυνατότητες τις εφαρμογής περιλαμβάνονται:

- Ρύθμιση της σειριακής επικοινωνίας Arduino – PC.
- Πλήρης έλεγχος του Arduino μέσω της σειριακής θύρας, δηλ. έλεγχος έναρξης και λήξης της δειγματοληψίας, χειροκίνητη δειγματοληψία, καθορισμός του ρυθμού δειγματοληψίας, αυτόματη ρύθμιση του διαιρέτη του ADC, ενημέρωση για το πλήθος και το είδος των συνδεδεμένων.
- Αποθήκευση των δεδομένων σε αρχείο κειμένου ή αυτόματη αποστολή σε φύλλο του EXCEL για περαιτέρω επεξεργασία.
- Ευέλικτος τρόπος γραφικής απεικόνισης των δεδομένων: ένα κανάλι ανά χρησιμοποιούμενο αισθητήρα, διαφορετικό χρώμα ανά κανάλι, δυνατότητα εμφάνισης ή απόκρυψης των σημείων ή των γραμμών μεταξύ τους, δυνατότητες ZoomIn, ZoomOut και ZoomOff ανά άξονα ή συνολικά.
- Εύχρηστο σύστημα τεσσάρων δρομέων (2 για τον άξονα χ και άλλοι 2 για τον y) επί της γραφικής παράστασης των δεδομένων με αυτόματη ενημέρωση κατάλληλου πίνακα για τις θέσεις τους ώστε να λαμβάνονται μετρήσεις κατευθείαν από τη γραφική παράσταση.

4. Επίδειξη πρωτοτύπου – Συμπεράσματα

Η διάταξη που παρουσιάζεται, προτείνεται για χρήση στο εργαστήριο Φυσικών Επιστημών των Γυμνασίων και Λυκείων. Ως εκ τούτου μέτρο σύγκρισης μπορεί να αποτελέσει το σύστημα Multilog με το οποίο είναι εφοδιασμένα τα περισσότερα εργαστήρια Φ.Ε. των Λυκείων. Καθώς η διάταξη βρίσκεται σε επίπεδο πρωτοτύπου είναι άκαιρη μία πλήρης σύγκριση των δύο συσκευών. Θα περιορίσουμε λοιπόν τη σύγκριση στα σημεία του πρωτοτύπου που έχουν λάβει την τελική τους μορφή:

- Το πρωτότυπό μας διαθέτει 3 εισόδους για αναλογικούς ή ψηφιακούς αισθητήρες, μία για ειδικού τύπου αισθητήρες και δύο στις οποίες μπορούν να συνδεθούν αισθητήρες που χρησιμοποιούν το πρωτόκολλο OneWire της Dallas Semiconductors (π.χ. θερμόμετρο DS18B20), έναντι τριών για αναλογικούς ή ψηφιακούς του Multilog που διπλασιάζονται με χρήση κατάλληλου διαχωριστικού καλωδίου (Multilog, 2003). Όμως η ύπαρξη θυρών επέκτασης για τους διαύλους I^2C και SPI, μιας πλήρους θύρας επέκτασης καθώς και η μεγέθους 32 kB μνήμη προγράμματος του Arduino αυξάνουν σημαντικά τις δυνατότητες του πρωτοτύπου. Μειονέκτημα είναι η έλλειψη δυνατότητας έναρξης ή λήξης της δειγματοληψίας μέσω

σκανδαλισμού.

- Τόσο το Multilog, όσο και το πρωτότυπό μας διαθέτουν ADC των 10bit, ενώ η χρήση ψηφιακού φίλτρου εξομάλυνσης μέσου όρου στο πρωτότυπό μας για χαμηλούς ρυθμούς δειγματοληψίας αυξάνει την ποιότητα των μετρήσεων. Δε χρησιμοποιήσαμε κάποιο άλλο φίλτρο καθώς η ισχυρή εξομάλυνση των κορυφών του σήματος με το φίλτρο κινούμενου μέσου όρου δεν ενδείκνυται για τις περιπτώσεις που απαιτείται μέτρηση του πλάτους του σήματος, ενώ για το φίλτρο διαμέσου θεωρήσαμε σημαντικότερη την ταχύτητα εκτέλεσης σε σχέση με τα οφέλη από τη χρήση του.
- Με χρήση μιας μόνο αναλογικής εισόδου, ο μέγιστος ρυθμός δειγματοληψίας του πρωτοτύπου μας είναι 2,5 kHz για γραφική απεικόνιση των δεδομένων σε πραγματικό χρόνο και 15,625 kHz για απεικόνιση μετά τη λήξη της λήψης. Οι αντίστοιχες τιμές για το Multilog (Multilog, 2003) είναι 100 Hz και 14,178 kHz. Για την επίτευξη αυτών των ρυθμών δειγματοληψίας στο πρωτότυπό μας χρησιμοποιήθηκε η δυνατότητα αλλαγής της συχνότητας χρονισμού του ADC χωρίς όμως σημαντική μείωση της ποιότητας των μετρήσεων. Πολλά υποσχόμενη είτε στον τομέα του ακόμη μεγαλύτερου ρυθμού δειγματοληψίας, είτε στη βελτίωση της ποιότητας των μετρήσεων κρίνεται η χρήση μνήμης SPI παρά του χρονικούς περιορισμούς που θέτει στη δειγματοληψία.
- Στο Multilog (Multilog, 2003) υπάρχουν χρονικοί περιορισμοί στη διάρκεια της δειγματοληψίας, ενώ στο πρωτότυπό μας ο μέγιστος αριθμός των μετρήσεων που μπορεί να ληφθούν περιορίζεται μόνο από το (έτσι κι αλλιώς πολύ μεγάλο) μέγεθος της μνήμης RAM του συστήματος απεικόνισης.
- Το Multilog (Multilog, 2003) έχει δυνατότητα αυτόματης αναγνώρισης και ρύθμισης των αισθητήρων, το πρωτότυπό μας όχι.

Λαμβανομένης υπόψη και της πολύ χαμηλής τιμής του Arduino θεωρούμε πως ένα ΣΣΛΑ βασισμένο στο Arduino αποτελεί μια πολύ καλή εναλλακτική πρόταση για την εκτέλεση μεγάλου αριθμού πειραμάτων Φυσικής, Χημείας και Βιολογίας στο εργαστήριο Φυσικών Επιστημών των Γυμνασίων και των Λυκείων. Ενδεικτικά αναφέρουμε πειράματα κινηματικής με αισθητήρα απόστασης ή φωτοπύλες, δυναμικής με αισθητήρα δύναμης, ηλεκτρισμού με αισθητήρες τάσης και έντασης ρεύματος, μελέτη οξέων βάσεων με αισθητήρα pH, μελέτη ενδόθερμων ή εξώθερμων αντιδράσεων με αισθητήρα θερμοκρασίας, κ.ά. Προϋπόθεση βέβαια αποτελεί η ανάπτυξη των κατάλληλων αισθητήρων ή άλλων συσκευών επέκτασης για το πρωτότυπό μας.

Αναφορές

Arduino (2013). Language Reference.

Ανάκτηση από το www.arduino.cc

Atmel (2013). ATmega48A/PA/88A/PA/168A/PA/328/P. Atmel Datasheet

AVR121 (2005). Enhancing ADC resolution by oversampling. Atmel Application Note.

Margolis, Michael (2012). Arduino Cookbook, Second Edition. Sebastopol: O' Reilly Media Inc.

Multilog (2003). Εγχειρίδιο χρήσης. Fourier Systems – Αμαξοτεχνική ΑΕΒΕ

OpenmusicLabs (2013). ATmega ADC.

Ανάκτηση από το www.openmusiclabs.com/learning/digital/atmega-adc

Putnam, William and R. Benjamin Knapp (1996). Input/Data Acquisition System Design for Human Computer Interfacing.

Ανάκτηση από το http://www.cs.princeton.edu/~prc/MUS539/Sensors.pdf

Wheat, Dale (2011). Arduino Internals. New York: Apress.

Abstract

In this project we investigate the capabilities of the Arduino platform in the field of acquiring data through digital and analog sensors. We emphasize on programming techniques aimed at improving measurement quality, increasing the sample rate as well as the baud rate of data to the output system. Lastly we present the prototype of a data acquisition system based on the Arduino platform.

Keywords: Arduino, sensor, analog to digital converter, sample rate, data acquisition system.