Σεμινάριο Ηλεκτρονικών 3-4-2014

Πρακτικές εφαρμογές με μικροελεγκτή

1η Εφαρμογή: Έλεγχος θερμοκρασίας

Μέτρηση θερμοκρασίας και on-off έλεγχος ανεμιστήρα με το λογισμικό LabView σε σειριακή επικοινωνία με το Arduino Uno.

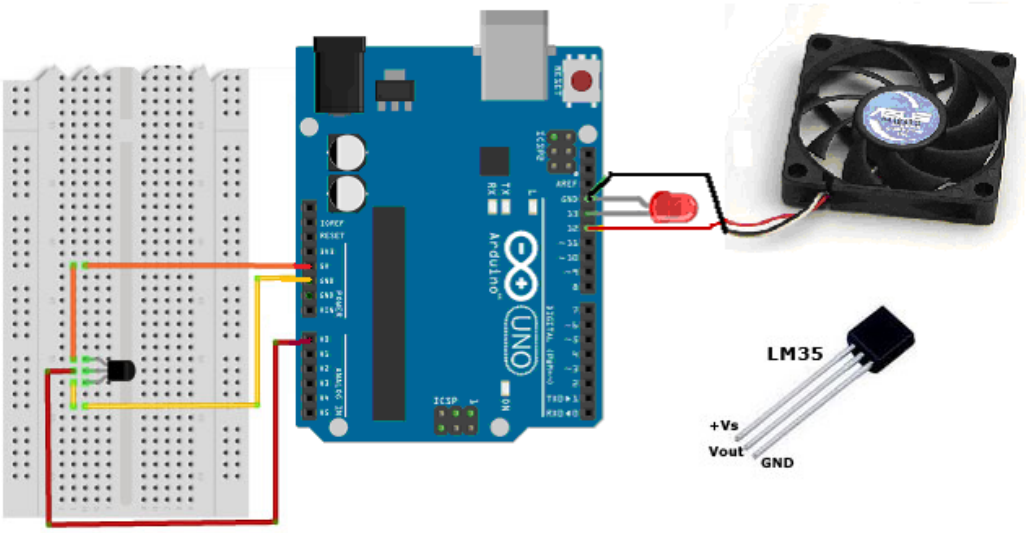
Ι. Μαλτέζος

Υλικά

Τα υλικά που χρειάστηκαν για αυτή την εφαρμογή, είναι:
-Ένα Arduino Uno
-Ένα Led
-Ένα μικρό ανεμιστηράκι
-USB καλώδιο
-Το λογισμικό Labview 7.0

Συνδεσμολογία με το Arduino

Για να συνδεθεί σωστά το κύκλωμα πρέπει να συνδεθεί το μακρύ ποδαράκι του Led στο pin13 και το κοντό ποδαράκι στη γείωση (GND). Συνήθως, οι περισσότερες πλακέτες έχουν ήδη ένα Led ενσωματωμένο. Επίσης συνδέουμε το θετικό (κόκκινο) καλώδιο του ανεμιστήρα στο pin12 και το αρνητικό (μαύρο) καλώδιο στη γείωση.

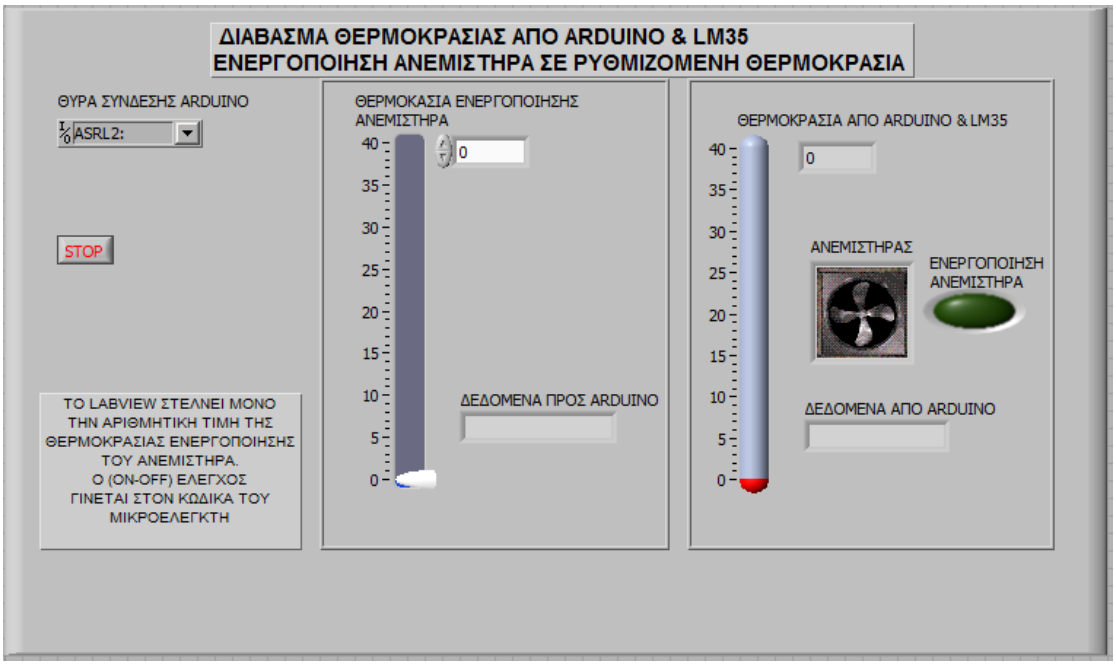


Πρόγραμμα

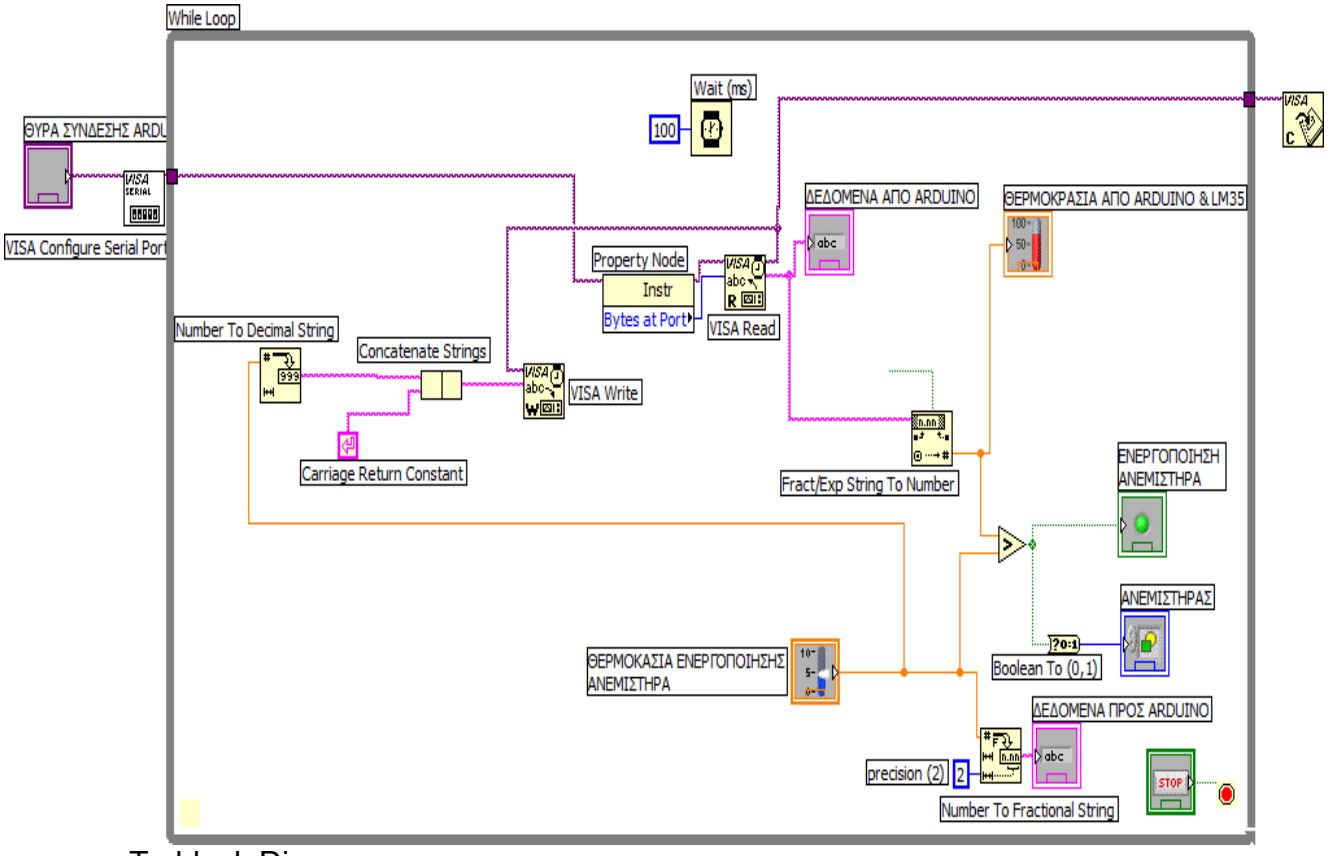
```
//----------ΟΡΙΣΜΟΣ ΜΕΤΑΒΛΗΤΩΝ----------------------------
float tempC; //ΟΡΙΣΜΟΣ ΜΕΤΑΒΛΗΤΗΣ ΤΥΠΟΥ FLOAT (ΔΕΚΑΔΙΚΟΥ ΑΡΙΘΜΟΥ) ΜΕ ΟΝΟΜΑ tempC ΜΕ ΣΚΟΠΟ ΝΑ ΧΡΗΣΙΜΟΠΟΙΗΘΕΙ ΑΠΟ ΤΟ ΠΡΟΓΡΑΜΜΑ ΓΙΑ ΤΗΝ ΑΠΟΘΗΚΕΥΣΗ ΤΗΣ ΤΙΜΗΣ ΤΗΣ ΘΕΡΜΟΚΡΑΣΙΑΣ
int sensorValue; //ΟΡΙΣΜΟΣ ΜΕΤΑΒΛΗΤΗΣ ΤΥΠΟΥ INTEGER (ΑΚΕΡΑΙΟΥ ΑΡΙΘΜΟΥ) ΜΕ ΟΝΟΜΑ sensorValue ΜΕ ΣΚΟΠΟ ΝΑ ΧΡΗΣΙΜΟΠΟΙΗΘΕΙ ΑΠΟ ΤΟ ΠΡΟΓΡΑΜΜΑ ΓΙΑ ΤΗΝ ΑΠΟΘΗΚΕΥΣΗ ΤΗΣ ΤΙΜΗΣ ΤΗΣ ΤΑΣΗΣ ΑΠΟ ΤΟΝ ΑΙΣΘΗΤΗΡΑ (0-1023)
int tempPin = 0; //ΟΡΙΣΜΟΣ ΜΕΤΑΒΛΗΤΗΣ ΤΥΠΟΥ INTEGER (ΑΚΕΡΑΙΟΥ ΑΡΙΘΜΟΥ) ΜΕ ΟΝΟΜΑ tempPin ΚΑΙ ΑΠΟΔΟΣΗ ΤΗΣ ΤΙΜΗΣ 0, ΜΕ ΣΚΟΠΟ ΝΑ ΧΡΗΣΙΜΟΠΟΙΗΘΕΙ ΑΠΟ ΤΟ ΠΡΟΓΡΑΜΜΑ ΓΙΑ ΤΗΝ ΕΠΙΛΟΓΗ ΤΟΥ ΚΑΝΑΛΙΟΥ ΕΙΣΟΔΟΥ ΤΟΥ ADC
int fanPin=12; //ΟΡΙΣΜΟΣ ΜΕΤΑΒΛΗΤΗΣ ΤΥΠΟΥ INTEGER (ΑΚΕΡΑΙΟΥ ΑΡΙΘΜΟΥ) ΜΕ ΟΝΟΜΑ fanPin ΚΑΙ ΑΠΟΔΟΣΗ ΤΗΣ ΤΙΜΗΣ 12, ΜΕ ΣΚΟΠΟ ΝΑ ΧΡΗΣΙΜΟΠΟΙΗΘΕΙ ΑΠΟ ΤΟ ΠΡΟΓΡΑΜΜΑ ΓΙΑ ΤΗΝ ΕΠΙΛΟΓΗ ΤΗΣ Ι/Ο ΘΥΡΑΣ ΣΥΝΔΕΣΗΣ ΤΟΥ ΑΝΕΝΙΣΤΗΡΑ
int ledPin=13; //ΟΡΙΣΜΟΣ ΜΕΤΑΒΛΗΤΗΣ ΤΥΠΟΥ INTEGER (ΑΚΕΡΑΙΟΥ ΑΡΙΘΜΟΥ) ΜΕ ΟΝΟΜΑ fanPin ΚΑΙ ΑΠΟΔΟΣΗ ΤΗΣ ΤΙΜΗΣ 13, ΜΕ ΣΚΟΠΟ ΝΑ ΧΡΗΣΙΜΟΠΟΙΗΘΕΙ ΑΠΟ ΤΟ ΠΡΟΓΡΑΜΜΑ ΓΙΑ ΤΗΝ ΕΠΙΛΟΓΗ ΤΗΣ Ι/Ο ΘΥΡΑΣ ΣΥΝΔΕΣΗΣ ΤΟΥ LED
float thermostate=0; //ΟΡΙΣΜΟΣ ΜΕΤΑΒΛΗΤΗΣ ΤΥΠΟΥ FLOAT (ΔΕΚΑΔΙΚΟΥ ΑΡΙΘΜΟΥ) ΜΕ ΟΝΟΜΑ thermostate ΚΑΙ ΑΠΟΔΟΣΗ ΤΗΣ ΑΡΧΙΚΗΣ ΤΙΜΗΣ 0, ΜΕ ΣΚΟΠΟ ΝΑ ΧΡΗΣΙΜΟΠΟΙΗΘΕΙ ΑΠΟ ΤΟ ΠΡΟΓΡΑΜΜΑ ΓΙΑ ΤΗΝ ΑΠΟΘΗΚΕΥΣΗ ΤΗΣ ΤΙΜΗΣ ΤΗΣ ΘΕΡΜΟΚΡΑΣΙΑΣ ΕΝΕΡΓΟΠΟΙΗΣΗΣ ΤΟΥ ΑΝΕΜΙΣΤΗΡΑ
//-------------------------------------------------------------

//----------ΑΡΧΙΚΕΣ ΡΥΘΜΙΣΕΙΣ ΜΙΚΡΟΕΛΕΓΚΤΗ-----------------
void setup() //ΣΥΝΑΡΤΗΣΗ ΑΡΧΙΚΩΝ ΡΥΘΜΙΣΕΩΝ ΤΟΥ ΜΙΚΡΟΕΛΕΓΚΤΗ
{
Serial.begin(9600); //ΑΡΧΙΚΟΠΟΙΗΣΗ ΤΗΣ ΣΕΙΡΙΑΚΗΣ ΕΠΙΚΟΙΝΩΝΙΑΣ ΜΕ ΤΑΧΥΤΗΤΑ 9600 bits/sec
analogReference(INTERNAL); //ΟΡΙΣΜΟΣ ΤΗΣ ΤΑΣΗΣ ΑΝΑΦΟΡΑΣ ΤΟΥ ADC ΣΤΑ 1,1 VOLTS ΜΕ ΧΡΗΣΗ ΕΣΩΤΕΡΙΚΗΣ ΣΤΑΘΜΗΣ ΑΝΑΦΟΡΑΣ ΤΟΥ ΜΙΚΡΟΕΛΕΓΚΤΗ.(ΜΕΙΩΣΗ ΑΠΟ ΤΑ 5 VOLTS ΣΤΑ 1,1 VOLTS ΓΙΑ ΚΑΛΥΤΕΡΗ ΔΙΑΚΡΙΤΙΚΗ ΑΚΡΙΒΕΙΑ - ΑΠΟ 4,9mV ΣΕ 1mV)
pinMode(fanPin, OUTPUT); //ΟΡΙΣΜΟΣ ΤΟΥ PIN ΜΕ ΑΡΙΘΜΟ fanPin ΔΗΛΑΔΗ ΤΟ 13 (ΟΠΩΣ ΟΡΙΣΤΙΚΕ ΠΑΡΑΠΑΝΩ ΣΤΙΣ ΜΕΤΑΒΛΗΤΕΣ)ΣΑΝ ΕΞΟΔΟΣ.
pinMode(ledPin, OUTPUT); //ΟΡΙΣΜΟΣ ΤΟΥ PIN ΜΕ ΑΡΙΘΜΟ ledPin ΔΗΛΑΔΗ ΤΟ 12 (ΟΠΩΣ ΟΡΙΣΤΙΚΕ ΠΑΡΑΠΑΝΩ ΣΤΙΣ ΜΕΤΑΒΛΗΤΕΣ)ΣΑΝ ΕΞΟΔΟΣ.
}



//-------------------------------------------------------------
```

```
void loop() //ΣΥΝΑΡΤΗΣΗ ΚΥΡΙΑΣ ΡΟΥΤΙΝΑΣ ΤΟΥ ΜΙΚΡΟΕΛΕΓΚΤΗ
{
//----ΔΙΑΒΑΣΜΑ ΜΕΤΡΗΣΕΩΝ ΘΕΡΜΟΚΡΑΣΙΑΣ ΑΠΟ ΤΟΝ ΑΙΣΘΗΤΗΡΑ-----
sensorValue = analogRead(tempPin); //ΔΙΑΒΑΣΜΑ ΤΗΣ ΤΑΣΗΣ ΠΟΥ ΔΙΝΕΙ ΤΟ LM35 ΑΠΟ ΤΟΝ ADC (ΚΑΝΑΛΙ ΕΙΣΟΔΟΥ 0 - PIN A0) ΚΑΙ ΑΠΟΔΟΣΗ ΤΗΣ ΤΙΜΗΣ (0-1023) ΣΤΗΝ ΜΕΤΑΒΛΗΤΗ sensorValue
tempC = sensorValue / 9.31; // ΔΙΑΙΡΕΣΗ ΤΗΣ ΤΙΜΗΣ ΠΟΥ ΔΙΑΒΑΣΤΗΚΕ ΜΕ ΤΟ 9.31(ΒΛ ΤΟ ΘΕΩΤΗΡΙΚΟ ΜΕΡΟΣ) ΓΙΑ ΤΗΝ ΜΕΤΑΤΡΟΠΗ ΣΕ ΒΑΘΜΟΥΣ ΚΕΛΣΙΟΥ ΚΑΙ ΑΠΟΔΟΣΗ ΤΗΣ ΤΙΜΗΣ ΣΤΗΝ ΜΕΤΑΒΛΗΤΗ tempC
//-------------------------------------------------------------

//-----ΑΠΟΣΤΟΛΗ ΜΕΤΡΗΣΕΩΝ ΘΕΡΜΟΚΡΑΣΙΑΣ ΣΤΟΝ ΥΠΟΛΟΓΙΣΤΗ------
Serial.println(tempC); // ΑΠΟΣΤΟΛΗ ΣΕΙΡΙΑΚΑ ΣΤΟΝ Η/Υ ΤΗΝ ΤΙΜΗ ΤΗΣ ΘΕΡΜΟΚΡΑΣΙΑΣ
delay(100); // ΚΑΘΥΣΤΕΡΗΣΗ 100ms (ΔΙΑΒΑΣΜΑ ΚΑΙ ΑΠΟΣΤΟΛΗ ΘΕΡΜΟΚΡΑΣΙΑΣ ΚΑΘΕ 100ms)
//-------------------------------------------------------------

//-----ΑΝΑΓΝΩΣΗ ΤΙΜΩΝ ΘΕΡΜΟΚΡΑΣΙΑΣ ΓΙΑ ΤΟΝ ΕΛΕΓΧΟ ΤΟΥ ΑΝΕΜΙΣΤΗΡΑ ΑΠΟ ΤΟΝ ΥΠΟΛΟΓΙΣΤΗ-----------------------------
thermostate = Serial.parseFloat(); // ΑΝΑΓΝΩΣΗ ΤΩΝ ΔΕΔΟΜΕΝΩΝ (ΣΕ ΜΟΡΦΗ ΧΑΡΑΚΤΗΡΩΝ - STRINGS) ΠΟΥ ΣΤΕΛΝΕΙ ΤΟ LABVIEW ΣΤΟ ARDUINO, ΑΝΙΧΝΕΥΣΗ ΤΩΝ ΑΡΙΘΜΗΤΙΚΩΝ ΔΕΔΟΜΕΝΩΝ, ΜΕΤΑΤΡΟΠΗ ΤΟΥΣ ΣΕ ΔΕΚΑΔΙΚΟΥΣ ΑΡΙΘΜΟΥΣ (FLOAT) ΚΑΙ ΑΠΟΘΗΚΕΥΣΗ ΤΟΥΣ ΣΤΗΝ ΜΕΤΑΒΛΗΤΗ thermostat
//-------------------------------------------------------------

//----------------(ON-OFF ΕΛΕΓΧΟΣ ΑΝΕΜΙΣΤΗΡΑ ΚΑΙ LED)-------

if (tempC < thermostate ) { // ΕΑΝ Η ΤΙΜΗ ΤΗΣ ΘΕΡΜΟΚΡΑΣΙΑΣ ΕΙΝΑΙ ΜΙΚΡΟΤΕΡΗ ΑΠΟ ΤΗΝ ΤΙΜΗ ΤΟΥ ΘΕΡΜΟΣΤΑΤΗ (ΑΥΤΗ ΠΟΥ ΟΡΙΖΕΤΑΙ ΣΤΟ LABVIEW) ΤΟΤΕ:
digitalWrite (fanPin, LOW); // ΒΓΑΛΕ ΛΟΓΙΚΟ ΜΗΔΕΝ ΣΤΟ PIN 12, ΔΗΛΑΔΗ ΔΕΝ ΕΧΟΥΜΕ ΕΝΕΡΓΟΠΟΙΗΣΗ ΤΟΥ ΑΝΕΜΙΣΤΗΡΑ
digitalWrite (ledPin, LOW); // ΒΓΑΛΕ ΛΟΓΙΚΟ ΜΗΔΕΝ ΣΤΟ PIN 13, ΔΗΛΑΔΗ ΔΕΝ ΕΧΟΥΜΕ ΕΝΕΡΓΟΠΟΙΗΣΗ ΤΟΥ LED
}
else //ΑΛΛΙΩΣ:
{
digitalWrite (fanPin, HIGH); //ΒΓΑΛΕ ΛΟΓΙΚΟ ΕΝΑ ΣΤΟ PIN 12, ΔΗΛΑΔΗ  ΕΧΟΥΜΕ ΕΝΕΡΓΟΠΟΙΗΣΗ ΤΟΥ ΑΝΕΜΙΣΤΗΡΑ
digitalWrite (ledPin, HIGH); //ΒΓΑΛΕ ΛΟΓΙΚΟ ΕΝΑ ΣΤΟ PIN 13, ΔΗΛΑΔΗ  ΕΧΟΥΜΕ ΕΝΕΡΓΟΠΟΙΗΣΗ ΤΟΥ LED
}
}
//-------------------------------------------------------------
```

Το πρόγραμμα στο Labview



To Front Panel



To block Diagram

Σχόλια-Ανάλυση του κώδικα

Αρχικά, ορίζουμε τις μεταβλητές του προγράμματος, όπου αν έχουμε δεκαδικές τιμές τις ορίζουμε σαν τύπος float, π.χ. η tempC για την απόδοση των τιμών της θερμοκρασίας.
Στις αρχικές ρυθμίσεις του μικροελεγκτή ορίζουμε την ταχύτητα της σειριακής επικοινωνίας, την τάση αναφοράς του ADC χρησιμοποιώντας μία εσωτερική τιμή τάσης αναφοράς 1.1 volts που διαθέτει ο Atmega328(αφού από το αισθητήριο δεν θα πάρουμε τάση μεγαλύτερης τιμής), και ορίζουμε επίσης τα pins12,13 σαν εξόδους.
Στην κύρια ρουτίνα του προγράμματος διαβάζουμε την τάση του αισθητηρίου με τον ADC και το αποτέλεσμα του διαιρούμε με 9.31 για να το μετατρέψουμε σε βαθμούς Κελσίου, έχοντας υπόψη μας τον ακόλουθο συλλογισμό:
Ορίζουμε τάση αναφοράς για τον ADC τα 1.1 volts. Ο ADC είναι 10 bits άρα έχουμε 2^10=1024 στάθμες με (1.1 / 1024)=1.0742mV διακριτική ικανότητα του ADC. Από το LM35 έχουμε 10mV ανά 1 βαθμό Κελσίου. Έτσι κάθε αλλαγή στην στάθμη του ADC (1.0742mV) αντιστοιχεί σε (1.0742 mV / 10 mV)=0.10742=(1 / 9.31) βαθμούς Κελσίου.
Στην συνέχεια στέλνουμε την τιμή της θερμοκρασίας στον Η/Υ, από όπου και διαβάζουμε ακόλουθα, την τιμή της θερμοκρασίας ενεργοποίησης του ανεμιστήρα.
Τέλος ακολουθεί το τμήμα ON-OFF ελέγχου του ανεμιστήρα.

Λίγα λόγια για το πρόγραμμα στο Labview

Με το **VISA Read** διαβάζουμε τα δεδομένα σε μορφή χαρακτήρων που στέλνει ο Arduino, όπου με το **Fract/Exp String To Number** τα μετατρέπουμε σε μορφή αριθμών για να τα απεικονίσουμε με το θερμόμετρο και να τα συγκρίνουμε με τα αριθμητικά δεδομένα που δημιουργεί ο ρυθμιστής θερμοκρασίας για το άναμα του Led στο Labview και την αλλαγή της εικόνας του ανεμιστήρα. Τα αριθμητικά αυτά δεδομένα του ρυθμιστή θερμοκρασίας με το **Number To Decimal String** τα μετατρέπουμε σε μορφή χαρακτήρων όπου με το **Concatenate Strings** τους προσθέτουμε τον χαρακτήρα “enter” (Τον χρειάζεται η συνάρτηση `Serial.parseFloat()` του Arduino για να ανιχνεύει το τέλος της μετάδοσης μίας τιμής) και τα στέλνουμε στον Arduino με το **VISA Write.**

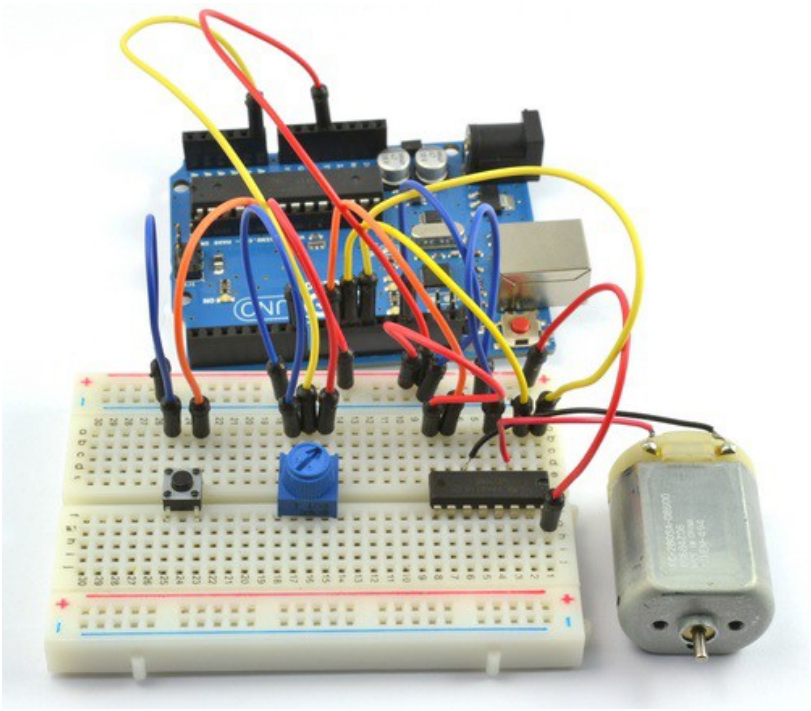
2η Εφαρμογή: Έλεγχος DC κινητήρα

Γιάννης Κουβαράκης

Έλεγχος DC Κινητήρα

με Arduino και το ολοκληρωμένο 293D

Created by Simon Monk



Επιμέλεια
Γιάννης Κουβαράκης
Απρίλιος 2014

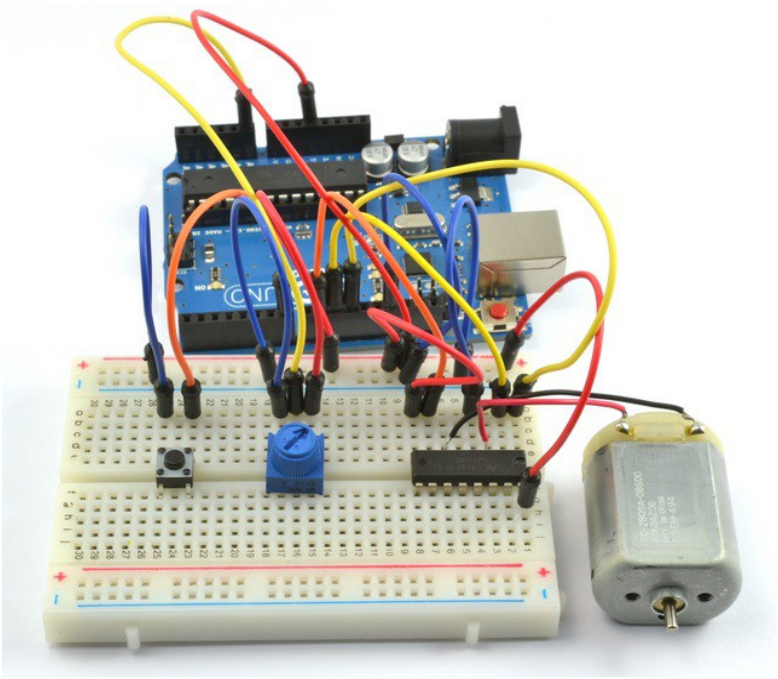
Πνευματικά δικαιώματα:

Πηγή για την εφαρμογή και την συγγραφή του φύλλου έργου που κρατάτε στα χέρια σας είναι η ιστοσελίδα της adafruit intustries: https://www.adafruit.com/
και δημιουργός του άρθρου ο Simon Monk

Adafruit Industries
150 VARICK ST
NEW YORK, NY 10013
Fax: (917) 210-3397

Εισαγωγή

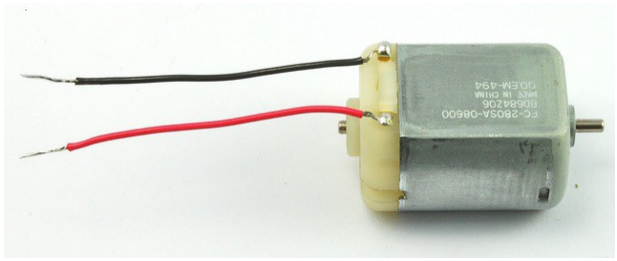
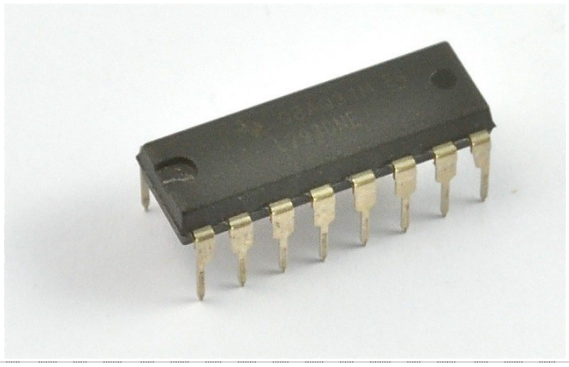
Σε αυτή την εφαρμογή, θα μάθουμε πως να ελέγχουμε την φορά περιστροφής και την ταχύτητα ενός μικρού DC κινητήρα χρησιμοποιώντας ένα Arduino και το L293D motor driver chip.

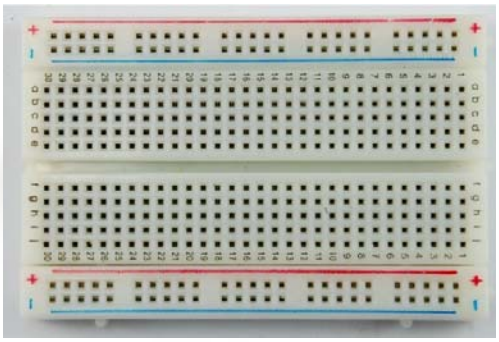
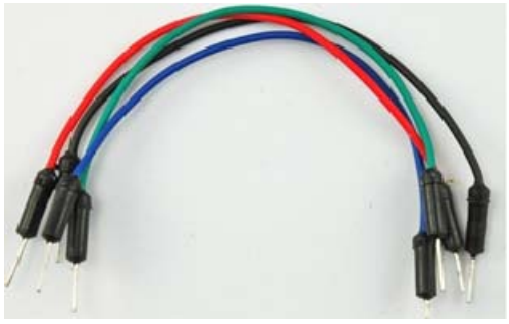


Στην εφαρμογή μας χρησιμοποιούμε ένα ποτενσιόμετρο για την ρύθμιση των στροφών και ένα push button για τον έλεγχο της φοράς περιστροφής του κινητήρα.

Εξαρτήματα-Υλικά

Για την εφαρμογή μας, θα χρησιμοποιήσουμε τα παρακάτω εξαρτήματα.

	Εξάρτημα	Ποσότητα
	Μικρός Κινητήρας 6V DC	1
	L293D IC	1
	10 kΩ μεταβλητή αντίσταση	1

	Διακόπτης push switch	1
	Μικρή Breadboard	1
	Arduino Uno R3	1
	καλώδια συνδεσμολογίας	16

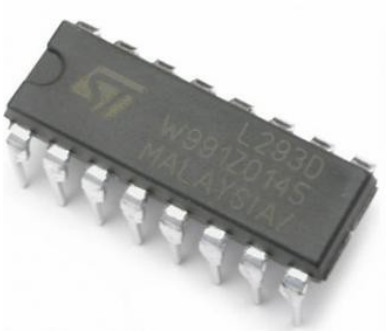
L293D

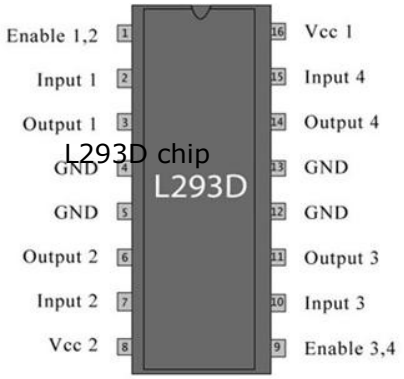
Ολοκληρωμένο κύκλωμα L293D

Το L293D είναι ένα 16pins chip και είναι μονολιθικό ολοκληρωμένο κύκλωμα, υψηλής τάσης. Αυτό σημαίνει ότι χρησιμοποιώντας αυτό το chip, μπορούν να χρησιμοποιηθούν DC κινητήρες και τροφοδοτικά μέχρι 36 Volts και παρέχει ένα μέγιστο ρεύμα 600mA ανά κανάλι.

Το ολοκληρωμένο είναι σχεδιασμένο για να ελέγχει 2 DC κινητήρες.
Αυτό γίνεται με τα 2 INPUT και 2 OUTPUT pins για τον κάθε κινητήρα.
Εμείς σε αυτό το πείραμα χρησιμοποιούμε το μισό μέρος (αριστερά όπως το βλέπουμε) του ολοκληρωμένου.

Τα περισσότερα pin του δεξιου μέρους είναι για τον δεύτερο κινητήρα.





L293D chip

Χαρακτηριστικά του ολοκληρωμένου L293D

Αριθμός pins	16
Τάση λειτουργίας	4.5V έως 36V
Μέγιστο ρεύμα ανά κανάλι	600mA
Input signals (Vi)	7V
Enable Voltage(Ven)	7V
Interface	TTL/CMOS
Θερμοκρασία λειτουργίας	-40 ~ 150°C

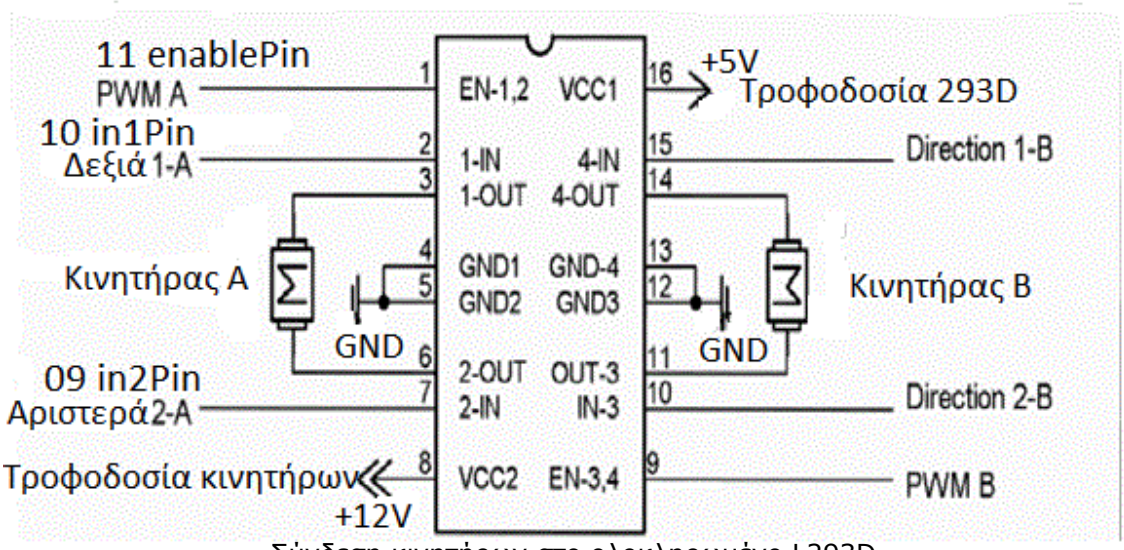
Λειτουργία του L293D

Για την λειτουργία του ολοκληρωμένου χρειάζεται είναι να ρυθμίσουμε σε κατάλληλα επίπεδα τους ακροδέκτες inputs του μικροελεγκτή για τον έλεγχο του κινητήρα και τα outputs σε κάθε ακροδέκτη του κινητήρα.

Αξίζει να αναφερθεί, ότι υπάρχουν επίσης δύο ακίδες 1 και 9 για ενεργοποίηση (Enable) στο ολοκληρωμένο, που αντιστοιχούν στους δύο κινητήρες και θα πρέπει να είναι σε κατάσταση high για να λειτουργήσει.

Όταν μια είσοδος enable είναι high το σχετικό pin ενεργοποιείται. Ως αποτέλεσμα, οι έξοδοι να καταστούν ενεργοί.

Ομοίως, όταν η είσοδος enable είναι low το σχετικό pin απενεργοποιείται, και το αποτέλεσμα, οι έξοδοι να καταστούν ανενεργοί



Σύνδεση κινητήρων στο ολοκληρωμένο L293D

Πρόκειται για ένα ολοκληρωμένο κύκλωμα το οποίο επιτρέπει την περιστροφή του κινητήρα και στις δύο κατευθύνσεις. Δηλαδή, μπορεί να προγραμματιστεί να ξεκινάει, να αντιστρέφει ή να σταματάει.

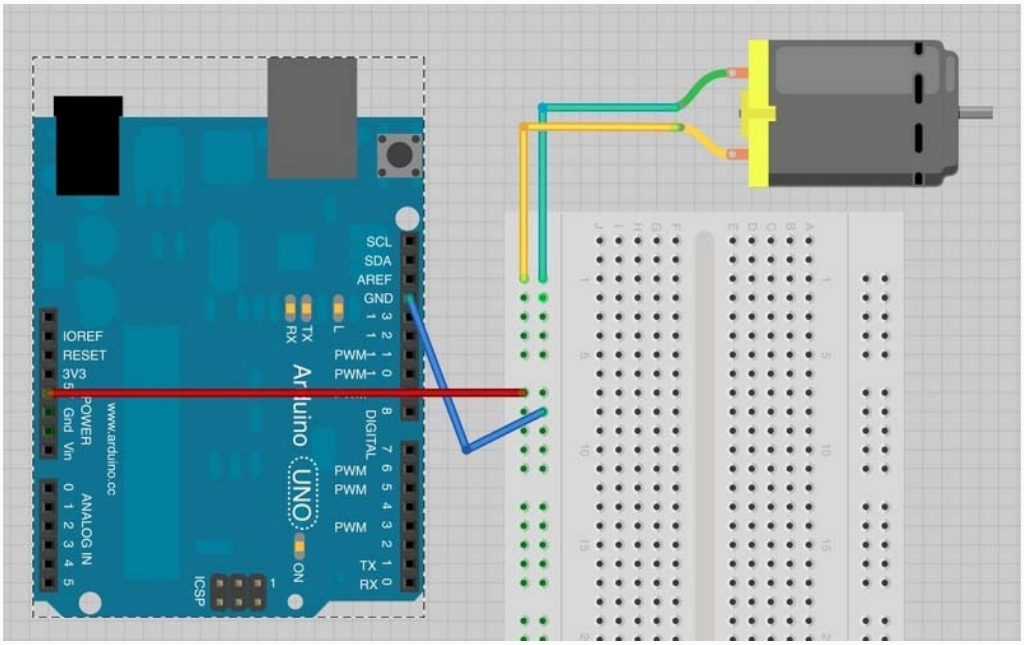
INT1	**INT2**	**Κινητήρας 1**
0	**1**	Περιστρέφεται σε μια κατεύθυνση
1	**0**	Περιστρέφεται σε αντίθετη κατεύθυνση

Παρόμοιος είναι και ο πίνακας αλήθειας για ένα άλλο κινητήρα που συνδέεται με OUT3 και OUT4 στο L293D και μπορεί να ελέγχεται μέσω INT3 και INT4

INT3	**INT4**	**Κινητήρας 2**
0	1	Περιστρέφεται σε μια κατεύθυνση
1	**0**	Περιστρέφεται σε αντίθετη κατεύθυνση

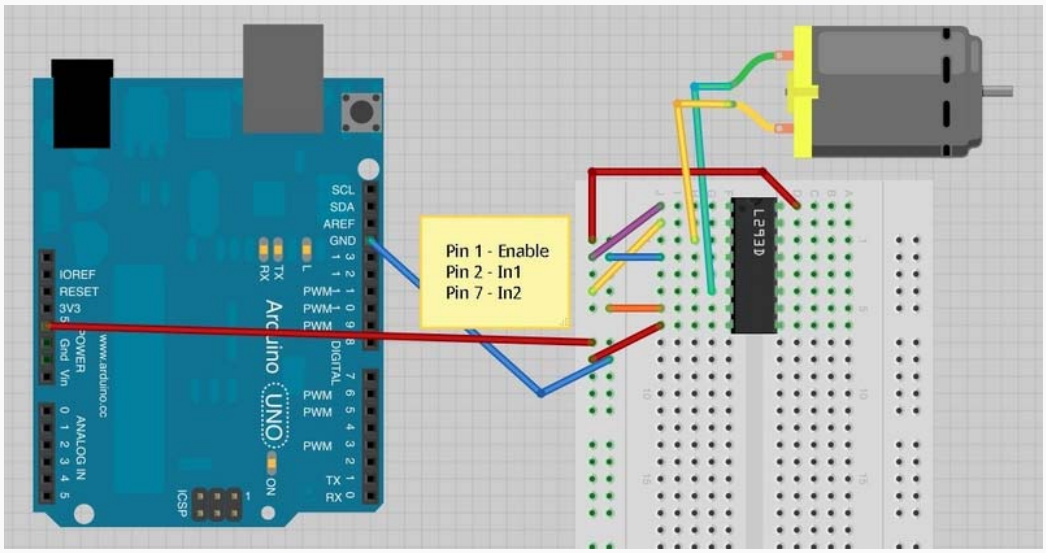
Εισαγωγική Εφαρμογή

Πριν πάρουμε την πλακέτα του Arduino για να ελέγξουμε τον κινητήρα, θα πειραματιστούμε με το L293D motor control chip για να κατανοήσουμε την βασική αρχή λειτουργίας του.
Μπορούμε να ξεκινήσουμε απλά χρησιμοποιώντας το Arduino για να παρέχει 5V στον κινητήρα.



Σημείωστε προς ποια πλευρά περιστρέφεται ο κινητήρας.
Μπορούμε να το κάνουμε αυτό πιάνοντας τον άξονα του κινητήρα ανάμεσα στα δάχτυλά μας.
Ανταλλάξτε τους ακροδέκτες του κινητήρα ώστε ο ακροδέκτης που πήγαινε στα +5V τώρα να πηγαίνει στην γείωση και αντίστροφα.
Ο κινητήρας τώρα θα περιστρέφεται προς την αντίθετη κατεύθυνση.
Αυτό μας δίνει μια ένδειξη για το πώς δουλεύει το τσιπάκι L293D. Οι ακροδέκτες ελέγχου του μας αφήνουν να κάνουμε κάτι ισοδύναμο της ανταλλαγής με τους ακροδέκτες του κινητήρα για να αντιστρέψουμε την κατεύθυνση.

Σύνδεστε την πλακέτα όπως φαίνεται παρακάτω. Το Arduino είναι ακόμα απλά για την παροχή ενέργειας, αλλά μπορούμε να πειραματιστούμε με τους ακροδέκτες ελέγχου πριν χρησιμοποιήσουμε το Arduino.



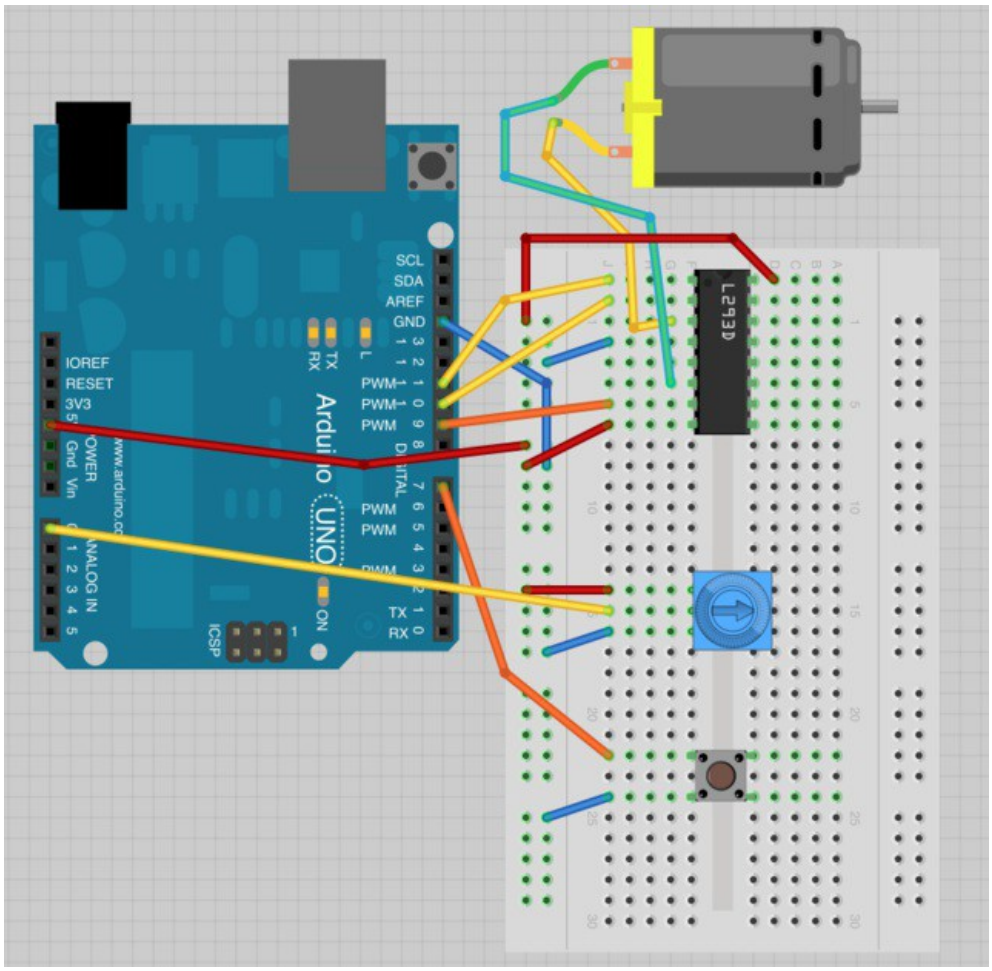
Οι τρεις ακροδέκτες του L293D που μας ενδιαφέρουν είναι ο ακροδέκτης 1 (Enable), ακροδέκτης 2 (In1) και ακροδέκτης 3 (In2).
Αυτοί είναι συνδεδεμένοι είτε στα 5V είτε στη γείωση χρησιμοποιώντας το μωβ, κίτρινο και πορτοκαλί καλώδιο βραχυκύκλωσης.
Όπως φαίνεται παραπάνω, ο κινητήρας θα πρέπει να στρέφεται προς μια κατεύθυνση, ας την ονομάσουμε κατεύθυνση A.
Αν μετακινήσουμε τον ακροδέκτη 1 (Enable) στην γείωση ο κινητήρας θα σταματήσει, ανεξάρτητα από το τι κάνουμε με τους ακροδέκτες ελέγχου In1 και In2.
Το enable ανάβει και σβήνει οτιδήποτε. Αυτό γίνεται χρήσιμο όταν χρησιμοποιούμε μια έξοδο PWM για να ελέγξουμε την ταχύτητα του κινητήρα.
Συνέδεουμε ξανά τον ακροδέκτη 1 στα 5V ώστε να ξεκινήσει ο κινητήρας.
Τώρα ας δοκίμασουμε να μετακινήσουμε το In1 (ακροδέκτης 2, κίτρινο καλώδιο) από τα 5V στη γείωση.
Αυτή τη στιγμή και το In1 και το In2 είναι συνδεδεμένα στη γείωση, έτσι πάλι ο κινητήρας θα σταματήσει.
Μετακινούμε το In2 από την γείωση στα 5V αυτό θα προκαλέσει στον κινητήρα περιστροφή προς την αντίθετη κατεύθυνση (κατεύθυνση B).
Τέλος, μετακίνηνούμε το In1 πίσω στα 5V έτσι ώστε και το In1 και το In2 να είναι στα 5V θα έχει σαν αποτέλεσμα να σταματήσει πάλι ο κινητήρας.
Η επίδραση από τους ακροδέκτες In1 και In2 του κινητήρα συνοψίζονται στον πίνακα από κάτω:

In1	In2	Κινητήρας
GND	GND	Σταματάει
5V	GND	Γυρνάει στην κατεύθυνση A
GND	5V	Γυρνάει στην κατεύθυνση B
5V	5V	Σταματάει

Συνδεσμολογία με το Arduino

Τώρα αφού κάναμε τα προηγούνα χειροκίνητα ας αφήσουμε τον Arduino να ελέγξει τα pins. Enable, In1 και In2

Όταν συνδέσουμε το ολοκληρωμένο πρέπει να σιγουρευτούμε ότι έχει τοποθετηθεί σωστά (με το notch στο επάνω μέρος)



Πρόγραμμα

Ας φορτώσουμε το παρακάτω sketch στον Arduino.

```
/*
Έλεγχος φοράς και ταχύτητας περιστροφής DC κινητήρα
*/

int enablePin = 11;
int in1Pin = 10;
int in2Pin = 9;
int switchPin = 7;
int potPin = 0;

void setup()
{
  pinMode(in1Pin, OUTPUT);
  pinMode(in2Pin, OUTPUT);
  pinMode(enablePin, OUTPUT);
  pinMode(switchPin, INPUT_PULLUP);
}

void loop()
{
  int speed = analogRead(potPin) / 4;
  boolean reverse = digitalRead(switchPin);
  setMotor(speed, reverse);
}

void setMotor(int speed, boolean reverse)
{
  analogWrite(enablePin, speed);
  digitalWrite(in1Pin, ! reverse);
  digitalWrite(in2Pin, reverse);
}
```

Σχόλια, ανάλυση του Κώδικα

```
/*
Έλεγχος φοράς και ταχύτητας περιστροφής DC κινητήρα
*/
int enablePin = 11;
//Ορίζουμε μια μεταβλητή τύπου integer (ακεραίου) με όνομα enablePin και της δίνουμετην τιμή 11 με σκοπό να την χρησιμοποιήσουμε παρακάτω στον κώδικα και να δηλώσουμε που θα συνδέσουμε τον ακροδέκτη 1 (enable 1) του chip 293D

int in1Pin = 10;
//Ορίζουμε μια μεταβλητή τύπου integer (ακεραίου) με όνομα in1Pin και της δίνουμετην τιμή 10 με σκοπό να την χρησιμοποιήσουμε παρακάτω στον κώδικα και να δηλώσουμε που θα συνδέσουμε τον ακροδέκτη 2 (input 1) του chip 293D

int in2Pin = 9;
//Ορίζουμε μια μεταβλητή τύπου integer (ακεραίου) με όνομα in2Pin και της δίνουμετην τιμή 9 με σκοπό να την χρησιμοποιήσουμε παρακάτω στον κώδικα και να δηλώσουμε που θα συνδέσουμε τον ακροδέκτη 7 (input 2) του chip 293D

int switchPin = 7;
//Ορίζουμε μια μεταβλητή τύπου integer (ακεραίου) με όνομα switchPin και της δίνουμετην τιμή 7 με σκοπό να την χρησιμοποιήσουμε παρακάτω στον κώδικα και να δηλώσουμε που θα συνδέσουμε τον ένα πόλο του διακόπτη αλλαγής φοράς περιστροφής του κινητήρα  (ο άλλος πόλος θα συνδεθεί στην γείωση)

int potPin = 0;
//Ορίζουμε μια μεταβλητή τύπου integer (ακεραίου) με όνομα potPin και της δίνουμε την τιμή 0 με σκοπό να την χρησιμοποιήσουμε παρακάτω στον κώδικα και να δηλώσουμε που θα συνδέσουμε τον δρομέα του ποτενσιόμετρου των 10ΚΩ ώστε να  ρυθμίσουμε τα 0 έως 5V στην αναλογική είσοδο A0 προκειμένου αφού μετατραπεί σε έναν αριθμό από 0-1023 να ρυθμίσουμε την ταχύτητα περιστροφής του κινητήρα

void setup()
//Η συνάρτηση αρχικών ρυθμίσεων του μικροεπεξεργαστή
{
```

```
pinMode(in1Pin, OUTPUT);
```
//Ορίζουμε χρησιμοποιώντας την μεταβλητή in1Pin ότι το pin 10 θα λειτουργεί σαν ψηφιακή έξοδος 0-5V ώστε να ενεργοποιούμε-απενεργοποιούμε τον ακροδέκτη 2 (input 1) του chip 293D

```
pinMode(in2Pin, OUTPUT);
```
//Ορίζουμε χρησιμοποιώντας την μεταβλητή in2Pin ότι το pin 9 θα λειτουργεί σαν ψηφιακή έξοδος 0-5V ώστε να ενεργοποιούμε-απενεργοποιούμε τον ακροδέκτη 7 (input 2) του chip 293D

```
pinMode(enablePin, OUTPUT);
```
//Ορίζουμε χρησιμοποιώντας την μεταβλητή enablePin ότι το pin 11 θα λειτουργεί σαν έξοδος 0-5V με PWM μεταβλητού Duty cycle* ώστε να ενεργοποιούμε-απενεργοποιούμε τον ακροδέκτη 1 (enable 1) του chip 293D με μεταβολή 0 έως 255

```
pinMode(switchPin, INPUT_PULLUP);
```
//Υπάρχουν pullup αντιστάσεις 20K ενσωματωμένες στο chip Atmega που μπορούν να ενεργοποιηθούν από το λογισμικό. Αυτές οι ενσωματωμένες αντιστάσεις pullup ενεργοποιούνται με την ρύθμιση του pinMode () ως INPUT_PULLUP. Εδώ με τον διακόπτη πατημένο έχουμε LOW ενώ με τον διακόπτη ανοιχτό HIGH

```
}
void loop()
```
//Η κύρια ρουτίνα του προγράμματος που εκτελείται συνεχώς
```
{
int speed = analogRead(potPin) / 4;
```
//Ορίζουμε μια μεταβλητή τύπου integer (ακεραίου) με όνομα speed να διαβάζει την αναλογική είσοδο διαιρώντας ταυτόχρονα δια 4 (0-1023/4=0-255) Ο συντελεστής της διαίρεσης είναι 4 γιατί ο μετατροπέας αναλογικού σε ψηφιακό στο A0 pin της αναλογικής εισόδου παίρνει τιμές μεταξύ 0 και 1023 και η αναλογική έξοδος στο pin 11 από 0 έως 255.

```
boolean reverse = digitalRead(switchPin);
```
// Εάν το button είναι πατημένο, ο κινητήρας γυρίζει δεξιόστροφα, διαφορετικά αριστερόστροφα. Η τιμή της μεταβλητής 'reverse' καθορίζεται από την θέση του διακόπτη (switchpin). Έχει δύο καταστάσεις ή θα είναι low (False), ή διαφορετικά θα είναι High (True).

```
setMotor(speed, reverse);
```

// Οι τιμές των μεταβλητών speed και reverse εισάγονται σε μια συνάρτηση που της δίνουμε το όνομα 'setMotor' έτσι ώστε να οδηγήσει τα κατάλληλα pins του driver chip ώστε να “ελεγχθεί” ο κινητήρας

}

```
void setMotor(int speed, boolean reverse)
```

// Η έκφραση void σημαίνει ότι η συνάρτηση δεν επιστρέφει κάποια τιμή (does not return any value). Δημιουργούμε-ορίζουμε μια συνάρτηση με int (ακέραιο μέρος) την μεταβλητή speed και Boolean (Λογικό High or Low) την μεταβλητή reverse

```
{
analogWrite(enablePin, speed);
```

// Δημιουργία παλμών PWM στο pin 11 με τιμή speed ώστε να διαβάζει την αναλογική είσοδο διαιρώντας ταυτόχρονα δια 4 (0-1023/4=0-255)

```
digitalWrite(in1Pin, ! reverse);
```

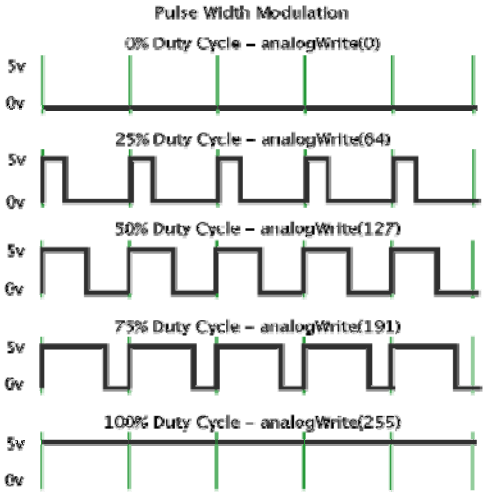
// έχουμε με παραπάνω εντολή ορίσει το pin 10 θα λειτουργεί σαν ψηφιακή έξοδος 0-5V μεταβλητή in1Pin έτσι ώστε να ενεργοποιούμε-απενεργοποιούμε τον ακροδέκτη 2 (input 1) του chip 293D (Το σύμβολο ! δηλώνει λογική άρνηση NOT)

```
digitalWrite(in2Pin, reverse);
```

// έχουμε με παραπάνω εντολή ορίσει το pin 9 θα λειτουργεί σαν ψηφιακή έξοδος 0-5V μεταβλητή in2Pin έτσι ώστε να ενεργοποιούμε-απενεργοποιούμε τον ακροδέκτη 7 (input 2) του chip 293D

```
}
```

*Duty cycle



Όπως εξηγήσαμε και πιο πάνω οι τιμές των μεταβλητών speed και reverse εισάγονται σε μια συνάρτηση που της δίνουμε το όνομα 'setMotor' έτσι ώστε να οδηγήσει τα κατάλληλα pins του driver chip ώστε να "ελεγχθεί" ο κινητήρας

```
void setMotor(int speed, boolean reverse)
{
  analogWrite(enablePin, speed);
  digitalWrite(in1Pin, !reverse);
  digitalWrite(in2Pin, reverse);
}
```

Πρώτα, η ταχύτητα ρυθμίζεται, με την χρήση της analogWrite στο enable pin. Το enable pin του L293 θέτει τον κινητήρα σε κατάσταση on ή off ανάλογα με την λογική στάθμη των pins in1 and in2 του L293.

Για τον έλεγχο της κατεύθυνσης του κινητήρα, τα pins in1 και in2 πρέπει να έχουν αντίθετες τιμές. Εάν το in1 είναι HIGH και το in2 είναι LOW, ο κινητήρας γυρίζει προς μία κατεύθυνση, σε αντίθετη περίπτωση σε αντίθετη κατεύθυνση.

Η εντολή '!' σημαίνει 'not'. Γιαυτό η πρώτη εντολή digitalWrite για το in1 το θέτει στο αντίστροφο της λογικής τιμής του 'reverse', αν το reverse είναι HIGH γίνεται LOW και αντίστροφα.

Η δεύτερη εντολή digitalWrite για το 'in2' θέτει το pin σε όποια τιμή είναι το 'reverse', αν το reverse είναι HIGH γίνεται HIGH και αντίστροφα.

Αυτό σημαίνει ότι τα δύο pin θα έχουν πάντα αντίθετες λογικές τιμές. Όταν το ένα θα είναι HIGH το άλλο θα είναι LOW και αντίστροφα

About the Author

Simon Monk is author of a number of books relating to Open Source Hardware.
The following books written by Simon are available from Adafruit:
Programming Arduino (http://adafru.it/1019),
30 Arduino Projects for the Evil Genius (http://adafru.it/868)
Programming the Raspberry Pi (http://adafru.it/aM5).

3η Εφαρμογή: **Μετρητής απόστασης με απεικόνιση της ένδειξης σε display**

Απεικόνιση χαρακτήρων σε display LCD και ενδείξεις από μετρητή απόστασης (distance sensor)

Χ. Δερμάτης

Υλικά

Τα υλικά που χρειάστηκαν για αυτή την εφαρμογή, είναι:
-Ένα Arduino Uno
-Ένα LCD
-Ένας αισθητήρας απόστασης HC-SR04
-USB καλώδιο

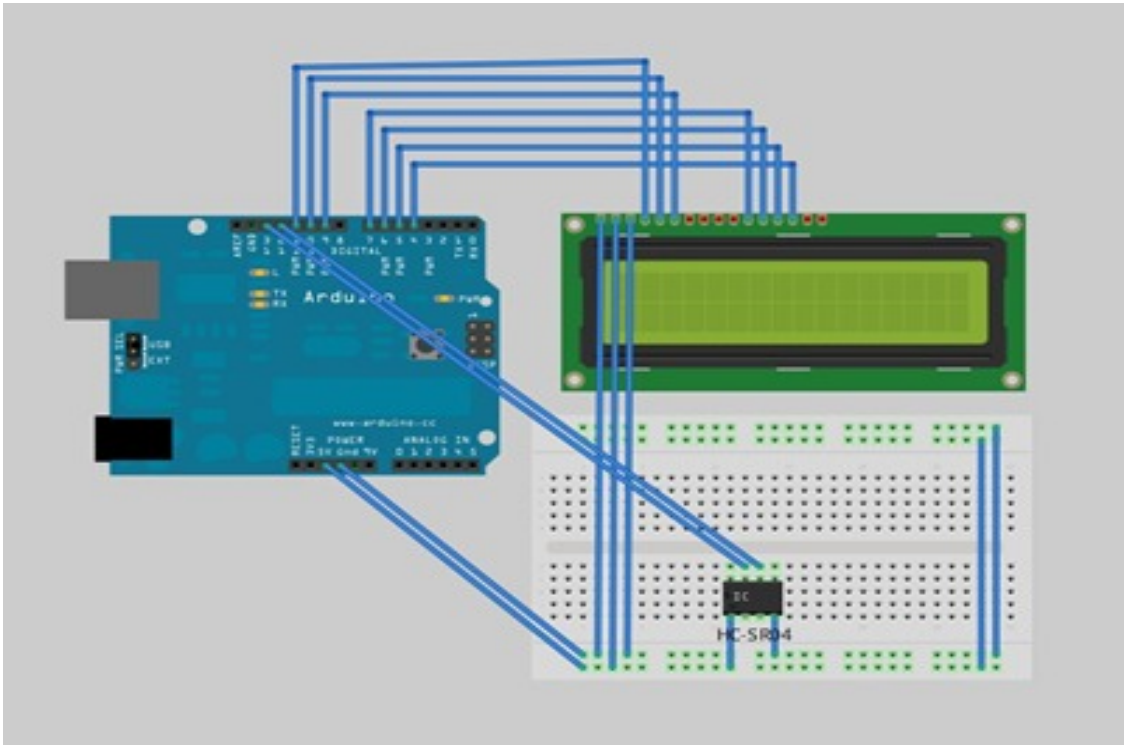
Εισαγωγή

Η αρχή λειτουργίας του αισθητήρα βασίζεται στην εκπομπή ενός σήματος και στη λήψη της ηχούς του για τον υπολογισμό της απόστασης. Η κατασκευή συνδυάζεται με τη πλατφόρμα του arduino και με μια οθόνη LCD στην οποία θα εμφανίζεται το αποτέλεσμα. Η προς μέτρηση απόσταση μπορεί να μεταβληθεί από 2 cm έως 400 cm.
Ο συγκεκριμένος αισθητήρας τροφοδοτείται από την πλακέτα του arduino με 5Volt.

Να δούμε πως μπορούμε να υπολογίσουμε την απόσταση του κάθε αντικειμένου μπροστά από τον αισθητήρα. Σε μία χρονική στιγμή t1 εκπέμπεται ένα σήμα και τη χρονική στιγμή παίρνουμε την ηχώ του. Το χρονικό διάστημα dt = t2 – t1 είναι ο συνολικός χρόνος που απαιτήθηκε για να φτάσει το σήμα στο αντικείμενο και να επιστρέψει. Όμως ο ακριβής χρόνος που χρειάστηκε για να φτάσει το σήμα στο αντικείμενο είναι dt/2. Η ταχύτητα του ήχου είναι 340.29 m / s και αν θέλουμε να το μετατρέψει σε εκατοστά ανά μsec το αποτέλεσμα είναι : 0.034049 CM / μsec. Η πραγματική απόσταση που διανύθηκε από τον ήχο ή την απόσταση μεταξύ του αισθητήρα και του αντικειμένου είναι: (dt / 2) * 0,034049

Συνδεσμολογία με το Arduino

Παρακάτω βλέπουμε την συνδεσμολογία της εφαρμογής .



Πρόγραμμα

```
#include <LiquidCrystal.h>

LiquidCrystal lcd( 7, 6, 5, 4, 3, 2);

int pingPin = 13; //ορίζουμε μια μεταβλητή τύπου ακεραίου με όνομα
                    pingPin και της δίνουμε την τιμή 13
int inPin = 12; //ορίζουμε μια μεταβλητή τύπου ακεραίου με όνομα
                  inPin και της δίνουμε την τιμή 12

void setup() // η συνάρτηση αρχικών ρυθμίσεων του μικροεπεξεργαστή
 lcd.begin(16, 2); // ορίζουμε στη οθόνης LCD τον αριθμό
                     στηλών και γραμμών
void loop() η κύρια υποροutίνα του προγράμματος που εκτελείται
            συνεχώς
 long duration, cm; // η απόσταση θα εκφραστεί σε cm
 pinMode(pingPin, OUTPUT);// ρυθμίζει το pin 13 ως έξοδο
 digitalWrite(pingPin, LOW); // ορίζουμε την ίδια έξοδο σε κατάσταση
                                off
 delayMicroseconds(2); // αναμονή για 2 μsec
 digitalWrite(pingPin, HIGH);//ορίζουμε την ίδια έξοδο σε on
                                κατάσταση
```

```
 delayMicroseconds(10); // αναμονή για 10 μsec
 digitalWrite(pingPin, LOW); // ρυθμίζει την ίδια έξοδο σε κατάσταση
                                off
 pinMode(inPin, INPUT); //ρυθμίζει το pin 12 ως είσοδο
 duration = pulseIn(inPin, HIGH);// μετέτρεψε το χρόνο σε απόσταση
 cm = microsecondsToCentimeters(duration);
 if (cm > 400) //εάν η απόσταση είναι μεγάλη (πιθανώς δεν μετρήθηκε
                 σωστά).
 lcd.clear(); //το αποτέλεσμα θα απορριφθεί
 lcd.setCursor(0,0);
 lcd.print("Nothing detected");
   }
 else { διαφορετικά

 clear lcd content  // καθάρισε το περιεχόμενο του lcd
 lcd.clear();
 lcd.setCursor(0, 0); //τοποθέτησε τον κέρσορα στη γραμμή 0 και στη
                        στήλη 0
 lcd.print("Object detected "); // τύπωσε το αντικείμενο ανιχνεύεται
 lcd.setCursor(0, 1); // τοποθέτησε  το κέρσορα στη στήλη 0 και στη
                       γραμμή 1
 lcd.print(cm);
 lcd.print("cm away!");
 delay(500);  // αναμονή για 0,5 sec
long microsecondsToCentimeters(long microseconds){
  // Η ταχύτητα του ήχου είναι 340m/s ή 29 μικροδευτερόλεπτα ανά
εκατοστό.
  // Το σήμα  ping εκπέμπεται και επιστρέφει έτσι ώστε να υπολογιστεί
η απόσταση του αντικειμένου.
  // Ως απόσταση παίρνουμε το μισό της απόστασης που διανύθηκε.
  return microseconds / 29 / 2;}
```

4η Εφαρμογή: **Πινακίδα οπτικών εφφέ** ()

Χρήση του Arduino για τον έλεγχο λειτουργίας πινακίδας LED.

Ν.Λεοντσίνης

Πινακίδα οπτικών εφφέ

*Χρήση του **Arduino** για τον έλεγχο λειτουργίας πινακίδας LED.*

ΥΛΙΚΑ ΠΟΥ ΧΡΕΙΑΖΟΜΑΣΤΕ:

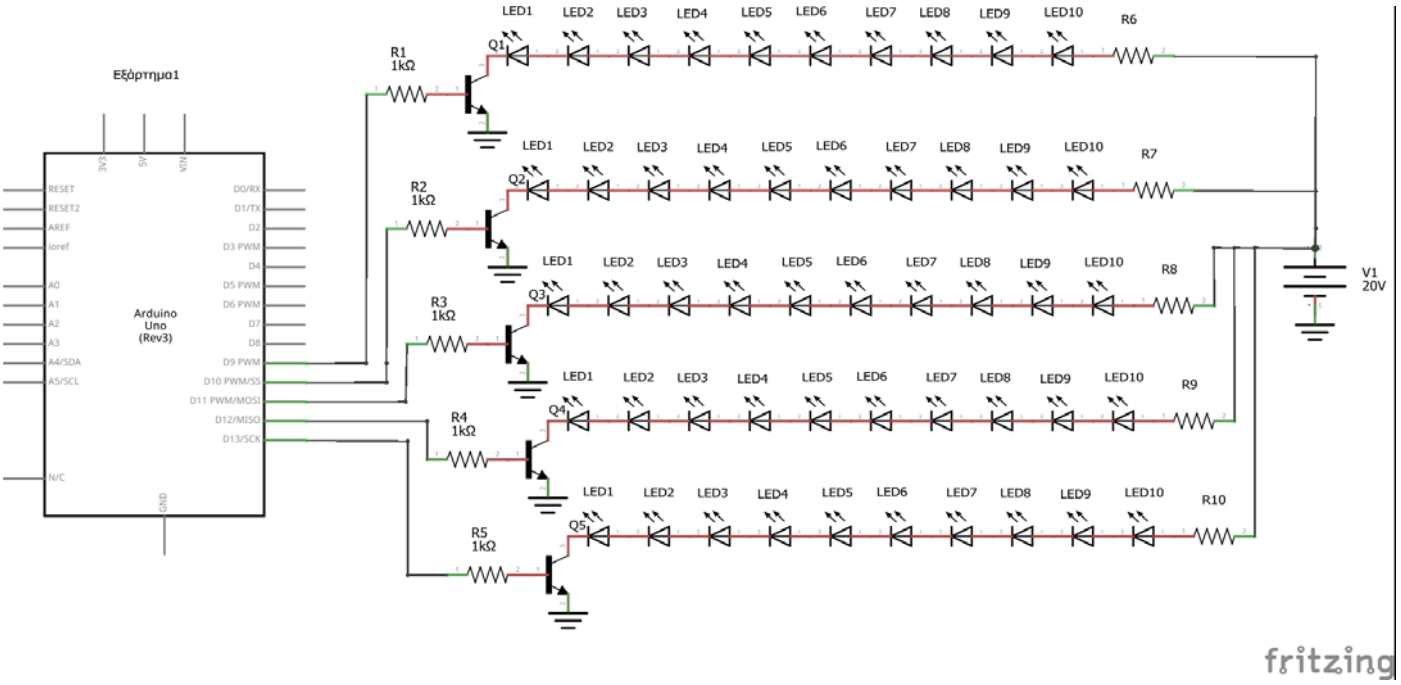
- ✓ *LED, ΑΝΤΙΣΤΑΣΕΙΣ, ΤΡΑΝΖΙΣΤΟΡ*
- ✓ *ΜΟΝΩΤΙΚΟ ΣΠΡΕΥ, ΘΕΡΜΟΚΟΛΛΑ, ΚΟΛΛΗΣΗ, ΠΛΑΚΕΤΑ ΓΕΝΙΚΗΣ ΧΡΗΣΗΣ*
- ✓ *ARDUINO*
- ✓ *ΤΡΟΦΟΔΟΤΙΚΟ*

Συλλάβαμε την ιδέα χρησιμοποίησης του ARDUINO τη σχολ. χρονιά 2012-2013, όταν οι μαθητές της B_{HN} τάξης του ΕΠΑΛ ΚΟΡΥΔΑΛΛΟΥ, στο μάθημα του PROJECT, αποφάσισαν να κατασκευάσουν μια πινακίδα με LED η οποία στην τελική της μορφή φαίνεται στην παρακάτω εικόνα.

Ο ARDUINO χρησιμοποιήθηκε για τον έλεγχο ανάμματος/σβησίματος των 5 πινακίδων.

ΣΥΝΔΕΣΜΟΛΟΓΙΑ ΜΕ ΤΟΝ ARDUINO

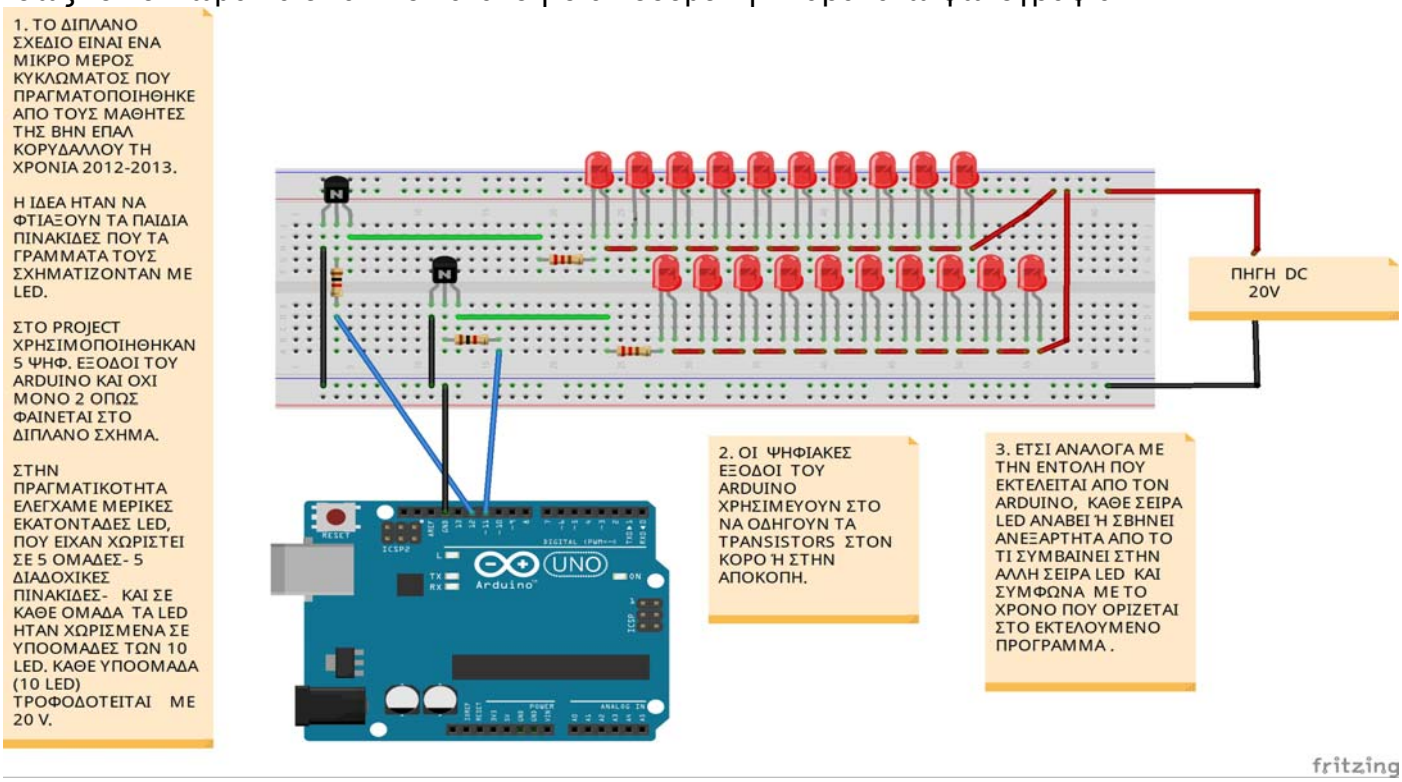
Για να απλοποιήσουμε το κύκλωμα, ας δούμε το παρακάτω:



Κάθε σειρά LED αντιστοιχεί σε μια από τις πέντε πινακίδες.

Ουσιαστικά, η μόνη διαφορά που έχει το πραγματικό κύκλωμα της πινακίδας με το παραπάνω σχεδιάγραμμα, είναι ότι σε κάθε πινακίδα, τα LED ήταν ομαδοποιημένα ανά 10 και οι 10/δες αυτές παράλληλα συνδεδεμένες μεταξύ τους.

Ίσως το κύκλωμα να είναι πιο κατανοητό αν δούμε την παρακάτω φωτογραφία.



Στην φωτογραφία εμφανίζονται οι 2 από τις 5 ομάδες των LED να έχουν συνδεθεί με δύο ψηφιακές εξόδους του ARDUINO.
Η συνδεσμολόγηση των άλλων τριών ομάδων είναι ακριβής επανάληψη αυτών .

ΠΡΟΓΡΑΜΜΑ ARDUINO

int led1 = 9; // Ορίζουμε μία μεταβλητή τύπου Integer (Ακεραίου) με όνομα led και της δίνουμε την τιμή 9 με σκοπό να την χρησιμοποιήσουμε παρακάτω στον κώδικα και να δηλώσουμε που θα συνδέσουμε το led.

```
int led2 = 10; //ΟΜΟΙΑ ΜΕ ΠΑΡΑΠΑΝΩ
int led3 = 11; //    -//-
int led4 = 12; //    -//-
int led5 = 13; //    -//-

// the setup routine runs once when you press reset:
void setup() {
  // ΑΡΧΙΚΟΠΟΙΗΣΗ  ΤΩΝ ΜΕΤΑΒΛΗΤΩΝ
  // ΟΡΙΖΕΙ ΠΟΙΑ PINS ΕΙΝΑΙ ΕΞΟΔΟΙ
  pinMode(led1, OUTPUT);     //PIN 9   ΕΞΟΔΟΣ
  pinMode(led2, OUTPUT);     //PIN 10 ΕΞΟΔΟΣ
  pinMode(led3, OUTPUT);     //PIN 11 ΕΞΟΔΟΣ
  pinMode(led4, OUTPUT);     //PIN 12 ΕΞΟΔΟΣ
  pinMode(led5, OUTPUT);     //PIN 13 ΕΞΟΔΟΣ
}

// the loop routine runs over and over again forever:
void loop() { //ΚΥΡΙΑ ΡΟΥΤΙΝΑ ΠΡΟΓΡΑΜΜΑΤΟΣ ΠΟΥ ΕΠΑΝΑΛΑΜΒΑΝΕΤΑΙ ΣΥΝΕΧΕΙΑ
  int i;
  digitalWrite(led1, LOW); //Η ΕΞΟΔΟΣ 9 ΣΕ ΚΑΤΑΣΤΑΣΗ LOW
  digitalWrite(led2, LOW);  //Η ΕΞΟΔΟΣ 10 ΣΕ ΚΑΤΑΣΤΑΣΗ LOW
  digitalWrite(led3, LOW);  //Η ΕΞΟΔΟΣ 11 ΣΕ ΚΑΤΑΣΤΑΣΗ LOW
  digitalWrite(led4, LOW);  //Η ΕΞΟΔΟΣ 12 ΣΕ ΚΑΤΑΣΤΑΣΗ LOW
  digitalWrite(led5, LOW);  //Η ΕΞΟΔΟΣ 13 ΣΕ ΚΑΤΑΣΤΑΣΗ LOW
  delay(1300);  // ΠΕΡΙΜΕΝΕ 1,3 SEC,

  digitalWrite(led5, HIGH); //Η ΕΞΟΔΟΣ 13 ΣΕ ΚΑΤΑΣΤΑΣΗ HIGH
  delay(1300); // ΠΕΡΙΜΕΝΕ 1,3 SEC,
  digitalWrite(led4, HIGH); //Η ΕΞΟΔΟΣ 12 ΣΕ ΚΑΤΑΣΤΑΣΗ HIGH
  delay(1300); // ΠΕΡΙΜΕΝΕ 1,3 SEC,
  digitalWrite(led3, HIGH); //Η ΕΞΟΔΟΣ 11 ΣΕ ΚΑΤΑΣΤΑΣΗ HIGH
  delay(1300); // ΠΕΡΙΜΕΝΕ 1,3 SEC,
  digitalWrite(led2, HIGH); //Η ΕΞΟΔΟΣ 10 ΣΕ ΚΑΤΑΣΤΑΣΗ HIGH
  delay(1300); // ΠΕΡΙΜΕΝΕ 1,3 SEC,
  digitalWrite(led1, HIGH); //Η ΕΞΟΔΟΣ 9 ΣΕ ΚΑΤΑΣΤΑΣΗ HIGH
  delay(1300); // ΠΕΡΙΜΕΝΕ 1,3 SEC,

  digitalWrite(led1, LOW); //Η ΕΞΟΔΟΣ 9 ΣΕ ΚΑΤΑΣΤΑΣΗ LOW
  digitalWrite(led2, LOW); //Η ΕΞΟΔΟΣ 10 ΣΕ ΚΑΤΑΣΤΑΣΗ LOW
  digitalWrite(led3, LOW); //Η ΕΞΟΔΟΣ 11 ΣΕ ΚΑΤΑΣΤΑΣΗ LOW
```

```
  digitalWrite(led4, LOW); //Η ΕΞΟΔΟΣ 12 ΣΕ ΚΑΤΑΣΤΑΣΗ LOW
  digitalWrite(led5, LOW); //Η ΕΞΟΔΟΣ 13 ΣΕ ΚΑΤΑΣΤΑΣΗ LOW
  delay(1000); // ΠΕΡΙΜΕΝΕ 1 SEC,


  for (i=0; i<5; i++) {//ΕΚΤΕΛΕΣΕ- ΕΠΑΝΕΛΑΒΕ 5 ΦΟΡΕΣ
// ΑΝΑΒΕΙ ΚΑΘΕ ΦΟΡΑ 1 ΠΙΝΑΚΙΔΑ
  digitalWrite(led1, LOW);
  digitalWrite(led2, HIGH);
  digitalWrite(led3, LOW);
  digitalWrite(led4, LOW);
  digitalWrite(led5, LOW);
  delay(100);
  digitalWrite(led1, LOW);
  digitalWrite(led2, LOW);
  digitalWrite(led3, LOW);
  digitalWrite(led4, HIGH);
  digitalWrite(led5, LOW);
  delay(100);
  digitalWrite(led1, LOW);
  digitalWrite(led2, LOW);
  digitalWrite(led3, LOW);
  digitalWrite(led4, LOW);
  digitalWrite(led5, HIGH);
  delay(100);
  digitalWrite(led1, HIGH);
  digitalWrite(led2, LOW);
  digitalWrite(led3, LOW);
  digitalWrite(led4, LOW);
  digitalWrite(led5, LOW);
  delay(100);
  digitalWrite(led1, LOW);
  digitalWrite(led2, LOW);
  digitalWrite(led3, HIGH);
  digitalWrite(led4, LOW);
  digitalWrite(led5, LOW);
  delay(500);
  }
     //ΑΝΑΒΟΥΝ ΟΛΕΣ ΟΙ ΠΙΝΑΚΙΔΕΣ
  digitalWrite(led1, HIGH);
  digitalWrite(led2, HIGH);
  digitalWrite(led3, HIGH);
  digitalWrite(led4, HIGH);
  digitalWrite(led5, HIGH);
  delay(1500);
   }
 // Η ΡΟΥΤΙΝΑ ΕΠΑΝΑΛΑΜΒΑΝΕΤΑΙ ΑΠΟ ΤΗΝ ΑΡΧΗ
```

ΣΧΟΛΙΑ - ΕΠΙΣΗΜΑΝΣΕΙΣ

Όπως γίνεται κατανοητό από το πρόγραμμα, χρησιμοποιούνται 5 ψηφιακές έξοδοι του ARDUINO τα pin 9, 10, 11, 12, 13.

Κάθε ψηφιακή έξοδος του ARDUINO σε κατάσταση LOW/HIGH δίνει 0V/5V, οπότε οδηγούνται τα TRA 2N2222 σε αποκοπή / κόρο παίζοντας το ρόλο του διακόπτη. Ο χρόνος κάθε φορά που η έξοδος παραμένει σε LOW ή HIGH κατάσταση καθορίζεται από την εντολή delay.

Κάθε φορά που το TRA οδηγείται σε αποκοπή, τα αντίστοιχα LED που είναι συνδεμένα με το TRA σβήνουν (έξοδος ARDUINO LOW) , ενώ κάθε φορά που το TRA οδηγείται σε κόρο τα αντίστοιχα LED ανάβουν (έξοδος ARDUINO HIGH).

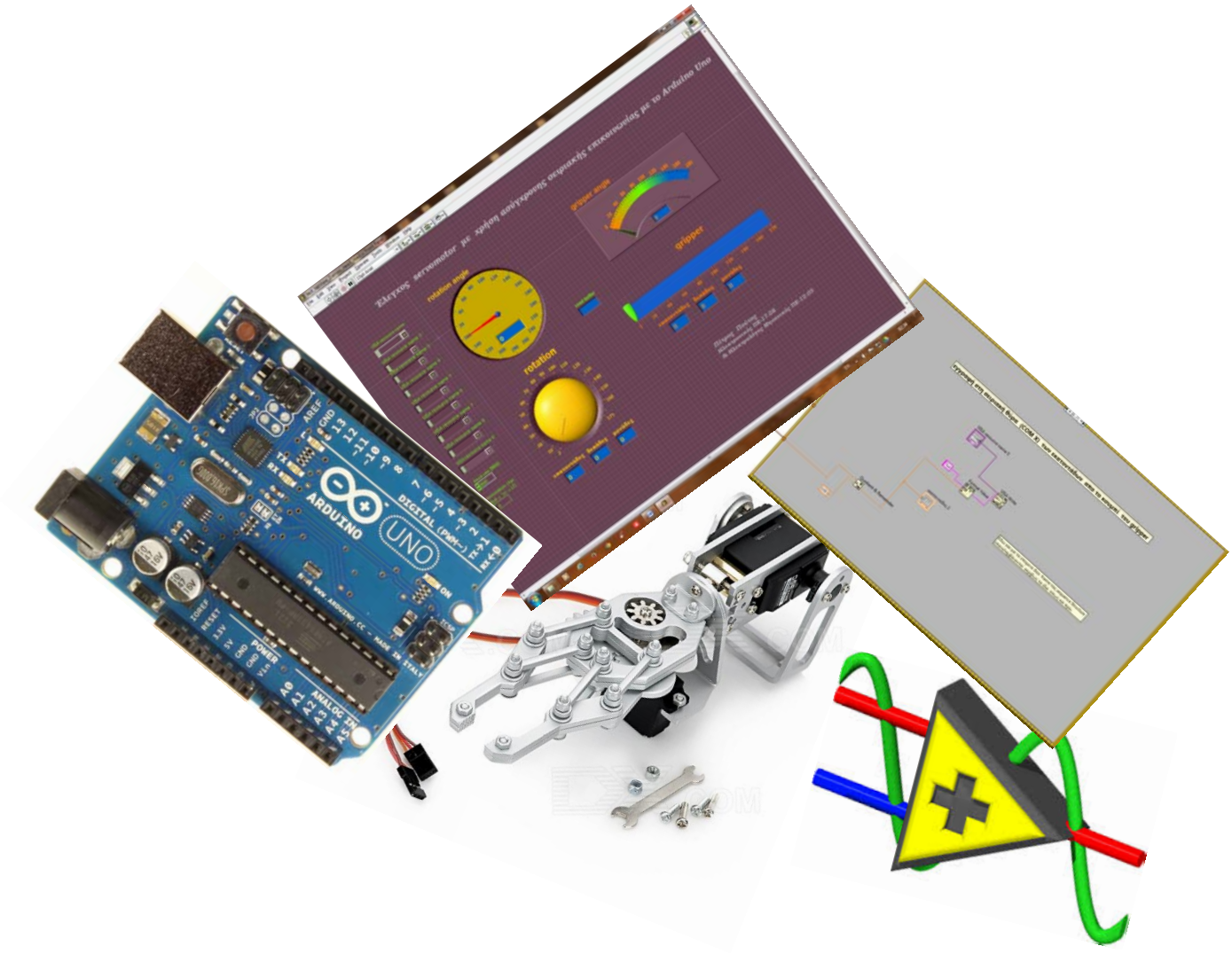
Με τη χρήση των TRA ξεπερνάμε τον περιορισμό των ψηφιακών εξόδων του ARDUINO που μπορούν να δώσουν έξοδο 5V/40-50mA.

Οι αντιστάσεις που χρησιμοποιούνται σε σειρά με τα LED όπως και η πηγή τροφοδοσίας των LED διαφέρουν κάθε φορά ανάλογα με το χρώμα και τη φωτεινότητα των LED που έχουμε επιλέξει.

5η Εφαρμογή: Έλεγχος κινητήρα servo -Ρομποτικός Βραχίονας

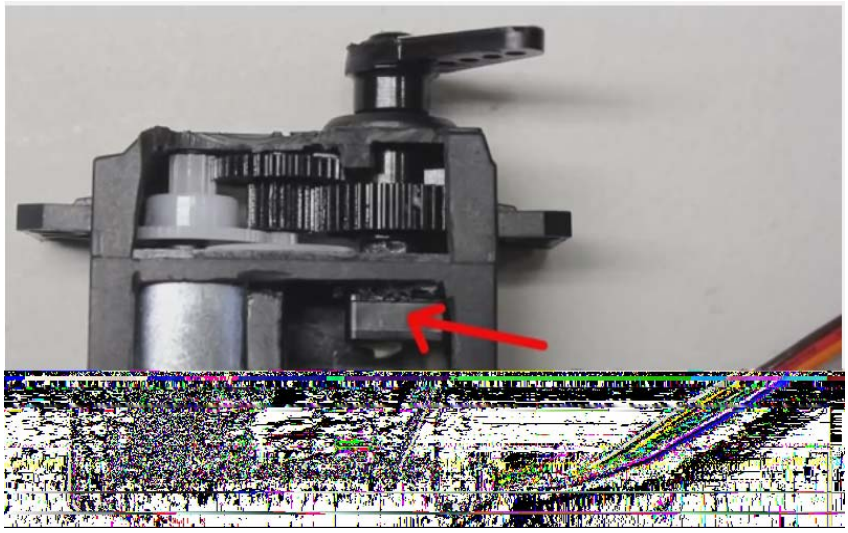
Έλεγχος κινητήρα servo (π.χ ενός ρομποτικού βραχίονα) χρησιμοποιώντας το λογισμικό LabView σε σειριακή επικοινωνία με το Arduino Uno

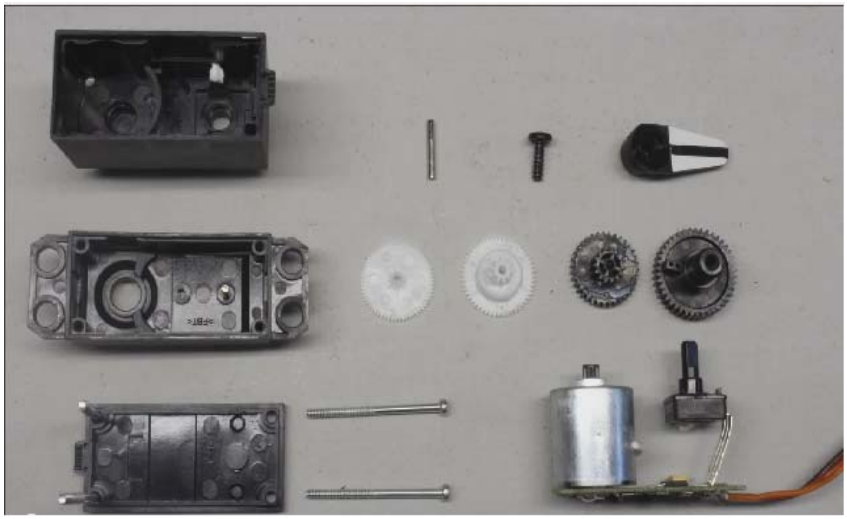
Π. Πούτος



Πληροφορίες για τα servomotors και για την ασύγχρονη σειριακή επικοινωνία του Arduino Uno (Atmega 328p) στη συγκεκριμένη εφαρμογή.

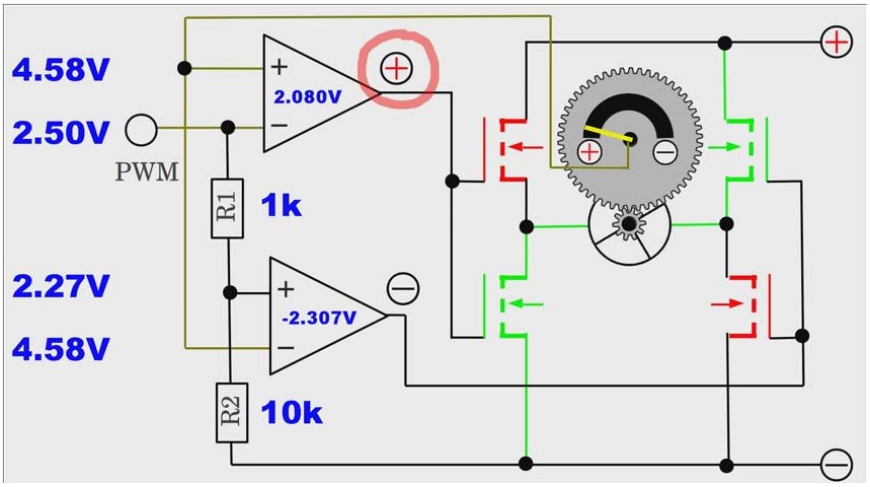
Α. *ΒΑΣΙΚΕΣ ΑΡΧΕΣ ΛΕΙΤΟΥΡΓΙΑΣ DC SERVOMOTORS (μικρής ισχύος με κινητήρα μόνιμου μαγνήτη)*

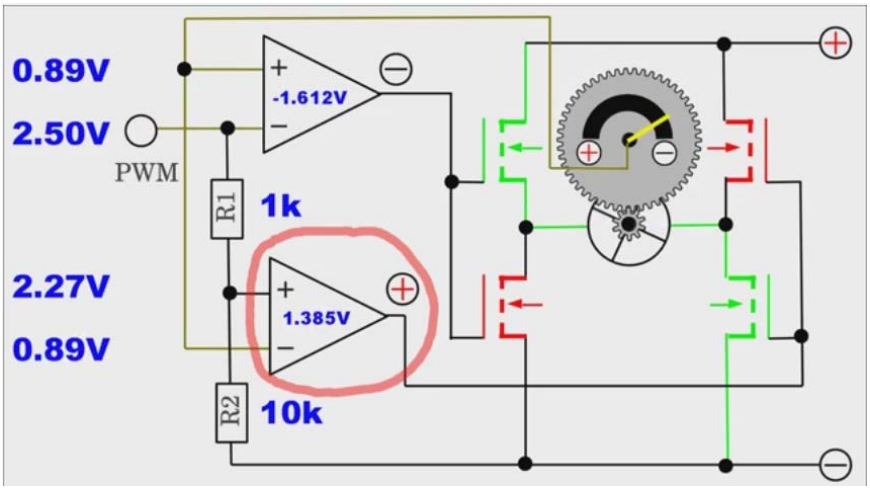


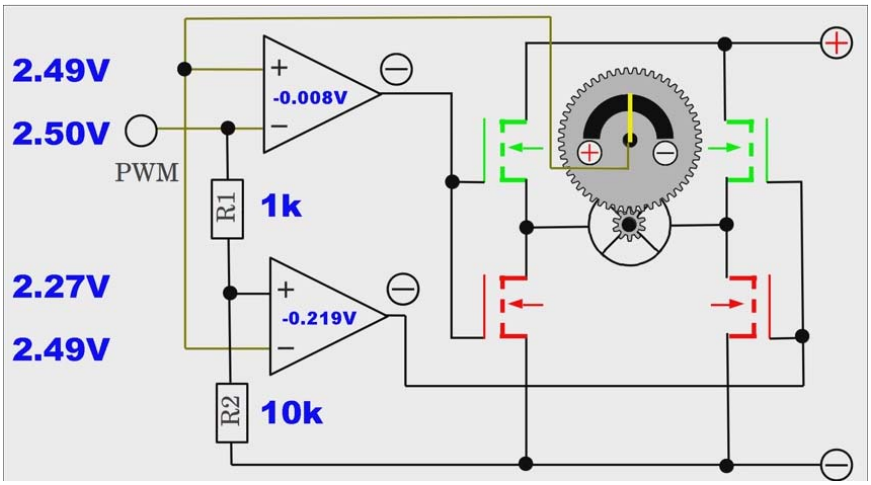


Στις φωτογραφίες, φαίνεται το εσωτερικό ενός μικρού servo σαν εκείνα που χρησιμοποιούνται στον μοντελισμό και στην «ερασιτεχνική ρομποτική», καθώς και τα επί μέρους εξαρτήματά του.
Αποτελείται από τον ηλεκτρικό κινητήρα συνεχούς ρεύματος με μόνιμο μαγνήτη στο στάτη , και τυλίγματα σε ποικίλο αριθμό πόλων. (3πολικά, 5πολικά κ.τ.λ), τον μηχανικό μειωτήρα (γρανάζια), που μας επιτρέπει να έχουμε μεγάλη ροπή με λίγες στροφές στον άξονα του servo, ένα ηλεκτρονικό κύκλωμα (servo driver) του οποίου η αρχή λειτουργίας φαίνεται κυκλωματικά παρακάτω, και ένα ποτενσιόμετρο που το στροφείο του είναι άμεσα συνδεδεμένο (μηχανικά) με τον άξονα του servo.

Έχει τρία καλώδια συνδεσμολογίας , με χρωματισμούς συνήθως καφέ , κόκκινο ,πορτοκαλί, ή μαύρο κόκκινο άσπρο. Το καφέ ή το μαύρο είναι η γείωση ή το - , το κόκκινο είναι το + και το πορτοκαλί ή το άσπρο είναι η τάση PWM που δέχεται .



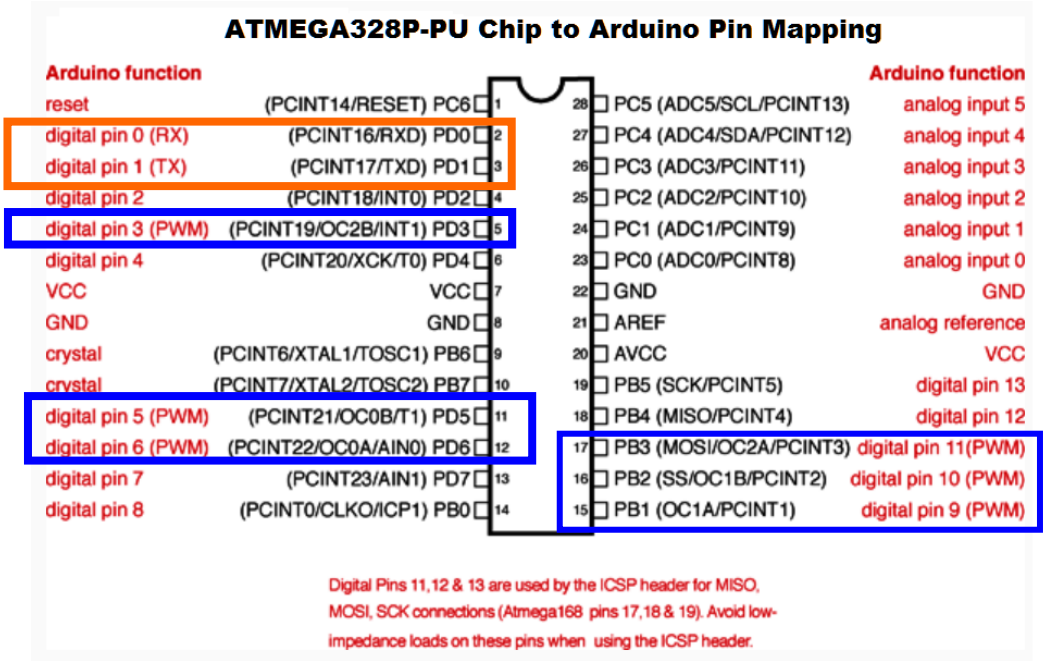




Στα κυκλωματικά διαγράμματα φαίνεται η λειτουργία του servo driver σε συνδυασμό με το ποτενσιόμετρο του άξονα.
Το servo driver αποτελείται από δύο συγκριτές τάσης , ένα μη αναστρέφοντα και ένα αναστρέφοντα , οι έξοδοι των οποίων είναι συνδεδεμένοι με τις πύλες των FET της Η γέφυρας . Στην αναστρέφουσα είσοδο του ενός και στην μη αναστρέφουσα του άλλου συγκριτή εφαρμόζεται η τάση PWM που στην ουσία είναι η μέση τιμή τάσης (DC) της παλμοσειράς PWM . Το μεσαίο ποδαράκι του ποτενσιόμετρου είναι συνδεδεμένο στις αντίστοιχες άλλες δύο εισόδους των συγκριτών . Τα δύο ακριανά ποδαράκια του ποτενσιόμετρου είναι συνδεδεμένα στην πηγή τροφοδοσίας του servo . Συνεχώς επιτελείται σύγκριση μεταξύ της PWM που στέλνουμε από το Arduino, και της τάσης από το ποτενσιόμετρο που όπως φαίνεται στα σχήματα, ορίζει τη θέση του άξονα του servo. Η περίοδος της PWM στην βιβλιοθήκη (Servo .h) είναι 20ms , αν δεν φτάνει παλμός κάθε 20ms τότε το servo παύει να λειτουργεί και είναι «έρμαιο» στο μηχανικό φορτίο του.

http://www.youtube.com/watch?v=v2jpnyKPH64 ***ίσως το πιο εκπαιδευτικό video για τις αρχές λειτουργίας ενός servo***

Β. ***Βασικές τεχνικές ασύγχρονης σειριακής επικοινωνίας μεταξύ Arduino Uno και Η/Υ μέσα από το λογισμικό LabView***



Τα pins **2 (Rx)** και **3 (Tx)** είναι τα default pins, που επικοινωνεί σειριακά το Arduino Uno, με οποιαδήποτε συσκευή εξωτερικά, σύμφωνα με το σειριακό πρότυπο διασύνδεσης RS-232C κάνοντας χρήση και ενός pin γείωσης, όπως φαίνεται και στο παρακάτω κυκλωματικό διάγραμμα.

Με προδιαγραφές : (8bits data, 1stop bit, parity None, baud rate από 300 μέχρι 115200) .

Τα data είναι σε ASCII κώδικα.

Hex	Dec	Char		Hex	Dec	Char	Hex	Dec	Char	Hex	Dec	Char
0x00	0	NULL	null	0x20	32	Space	0x40	64	@	0x60	96	`
0x01	1	SOH	Start of heading	0x21	33	!	0x41	65	A	0x61	97	a
0x02	2	STX	Start of text	0x22	34	"	0x42	66	B	0x62	98	b
0x03	3	ETX	End of text	0x23	35	#	0x43	67	C	0x63	99	c
0x04	4	EOT	End of transmission	0x24	36	$	0x44	68	D	0x64	100	d
0x05	5	ENQ	Enquiry	0x25	37	%	0x45	69	E	0x65	101	e
0x06	6	ACK	Acknowledge	0x26	38	&	0x46	70	F	0x66	102	f
0x07	7	BELL	Bell	0x27	39	'	0x47	71	G	0x67	103	g
0x08	8	BS	Backspace	0x28	40	(	0x48	72	H	0x68	104	h
0x09	9	TAB	Horizontal tab	0x29	41	)	0x49	73	I	0x69	105	i
0x0A	10	LF	New line	0x2A	42	*	0x4A	74	J	0x6A	106	j
0x0B	11	VT	Vertical tab	0x2B	43	+	0x4B	75	K	0x6B	107	k
0x0C	12	FF	Form Feed	0x2C	44	,	0x4C	76	L	0x6C	108	l
0x0D	13	CR	Carriage return	0x2D	45	-	0x4D	77	M	0x6D	109	m
0x0E	14	SO	Shift out	0x2E	46	.	0x4E	78	N	0x6E	110	n
0x0F	15	SI	Shift in	0x2F	47	/	0x4F	79	O	0x6F	111	o
0x10	16	DLE	Data link escape	0x30	48	0	0x50	80	P	0x70	112	p
0x11	17	DC1	Device control 1	0x31	49	1	0x51	81	Q	0x71	113	q
0x12	18	DC2	Device control 2	0x32	50	2	0x52	82	R	0x72	114	r
0x13	19	DC3	Device control 3	0x33	51	3	0x53	83	S	0x73	115	s
0x14	20	DC4	Device control 4	0x34	52	4	0x54	84	T	0x74	116	t
0x15	21	NAK	Negative ack	0x35	53	5	0x55	85	U	0x75	117	u
0x16	22	SYN	Synchronous idle	0x36	54	6	0x56	86	V	0x76	118	v
0x17	23	ETB	End transmission block	0x37	55	7	0x57	87	W	0x77	119	w
0x18	24	CAN	Cancel	0x38	56	8	0x58	88	X	0x78	120	x
0x19	25	EM	End of medium	0x39	57	9	0x59	89	Y	0x79	121	y
0x1A	26	SUB	Substitute	0x3A	58	:	0x5A	90	Z	0x7A	122	z
0x1B	27	FSC	Escape	0x3B	59	;	0x5B	91	[	0x7B	123	{
0x1C	28	FS	File separator	0x3C	60	<	0x5C	92	\	0x7C	124	|
0x1D	29	GS	Group separator	0x3D	61	=	0x5D	93	]	0x7D	125	}
0x1E	30	RS	Record separator	0x3E	62	>	0x5E	94	^	0x7E	126	~
0x1F	31	US	Unit separator	0x3F	63	?	0x5F	95	_	0x7F	127	DEL

ASCII table

<u>Το ζητούμενο σε αυτό το project είναι:</u> *Από δύο εικονικά ποτενσιόμετρα μέσα από το περιβάλλον του LabView να ελέγχουμε (κινούμε) δύο servo motors με ακρίβεια 1°. Το interface είναι ο μ/ε atmega 328p, που προγραμματίζεται με τη πλατφόρμα Arduino.*

<u>Στη συγκεκριμένη εφαρμογή</u> πρέπει να στέλνουμε από την σειριακή θύρα ενός Η/Υ, προς τη σειριακή θύρα του Arduino Uno, ***αριθμητικές ποσότητες*** που ουσιαστικά αντιπροσωπεύουν την γωνία περιστροφής του άξονα του servomotor.

Από την σειριακή θύρα όμως του Η/Υ, στέλνονται ASCII χαρακτήρες χωρίς φυσικά ***αριθμητικό βάρος,*** και έτσι τις λαμβάνουμε και από τη σειριακή θύρα του Arduino Uno .

<u>Το πρόβλημα συνεπώς, είναι να μετατρέψουμε τους χαρακτήρες αυτούς σε αριθμητικές ποσότητες , και συγκεκριμένα σε αριθμητικό δεκαδικό ακέραιο.</u>

Το παράδειγμα που ακολουθεί είναι αντιπροσωπευτικό για την διαδικασία που χρησιμοποιείται για την σειριακή επικοινωνία μεταξύ Η/Υ (π.χ μέσω του λογισμικού LabView) και του Arduino Uno.

Έστω ότι από το «κουμπί» του ποτενσιόμετρου **rotation** στο frond panel του LabView στέλνουμε την τιμή 132 που αντιπροσωπεύει τις μοίρες που θέλουμε να στρέψουμε τον άξονα του servo. Ο αριθμός αυτός , σε ASCII κώδικα θα «*φύγει*» από τη σειριακή θύρα σαν
‘1’ → (49 dec) , ‘3’ → (51 dec) και ‘2’ → (50 dec) σύμφωνα με τον πίνακα συμβόλων ASCII , τα οποία δεν είναι οι αριθμητικές ποσότητες που θέλουμε.

Στην **ανάγνωση** (*Serial.Read*) της σειριακής θύρας του Arduino, από κάθε δεκαδικό αριθμό που αντιπροσωπεύει το ‘1’ , ‘3’ , ‘2’ , αφαιρούμε το ‘0’ → (48 dec).

Έτσι έχουμε 49-48 =1 , 51-48 =3 και 50-48 =2 . Συνεπώς έχουμε τα δεκαδικά ψηφία που σχηματίζουν τον επιθυμητό αριθμό 132 , αλλά χωρίς βάρη . Έτσι πολλαπλασιάζουμε με τα ανάλογα δεκαδικά βάρη κάθε δεκαδικό ψηφίο , 1Χ 100 = 100 , 3Χ10= 30 , 2Χ1=2 , και τέλος αθροίζουμε 100+30+2 =132 κι έτσι έχουμε την δεκαδική ποσότητα 132.

Αυτή τη ποσότητα (ακέραια μεταβλητή: int rotation) την γράφουμε στο pin του Arduino που οδηγεί το servo. (π.χ **myservo1**.write(rotation).

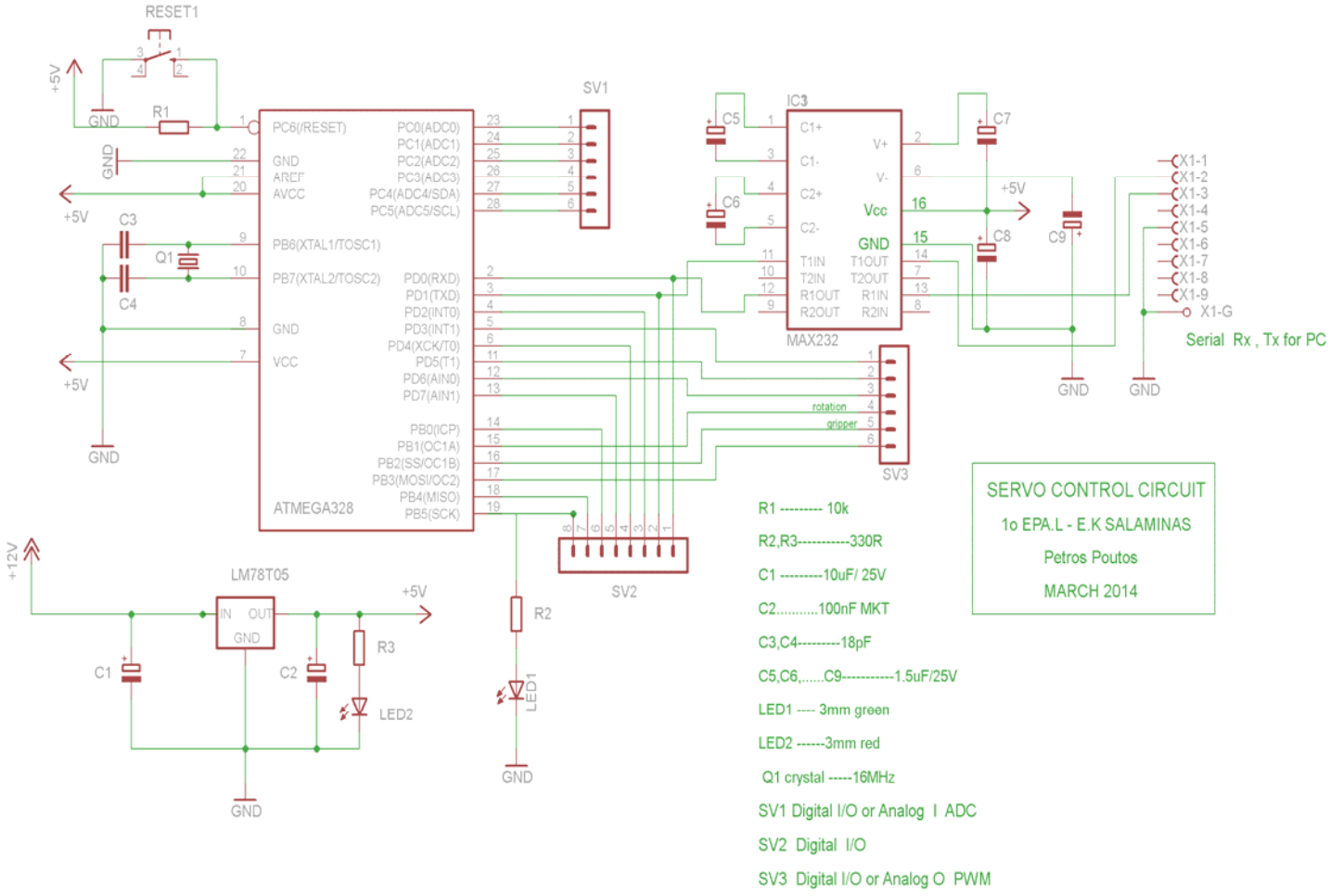
Οι δεκαδικές ποσότητες που γράφουμε στα pin του Arduino που οδηγούν τα servo, αντιπροσωπεύουν μοίρες στροφής του άξονα. Στην πραγματικότητα , στην βιβλιοθήκη που χρησιμοποιούμε (Servo.h), και μάλιστα με την εντολή *attach*.( ) έχει γίνει αντιστοίχιση (mapping) των μοιρών με την διάρκεια σε μs του ενεργού παλμού στην PWM κυματομορφή , που βγάζει το αντίστοιχο pin.

Συγκεκριμένα στην default κατάσταση της *attach* .( ) έχουμε για 0° → 544μs και για 180° ή όπου αλλού *τερματίζει* το servo → 2400μs, είναι φανερό ότι στα 928μs ο άξονας του servo θα βρίσκεται στο μέσον της διαδρομής του.(π.χ 90°)

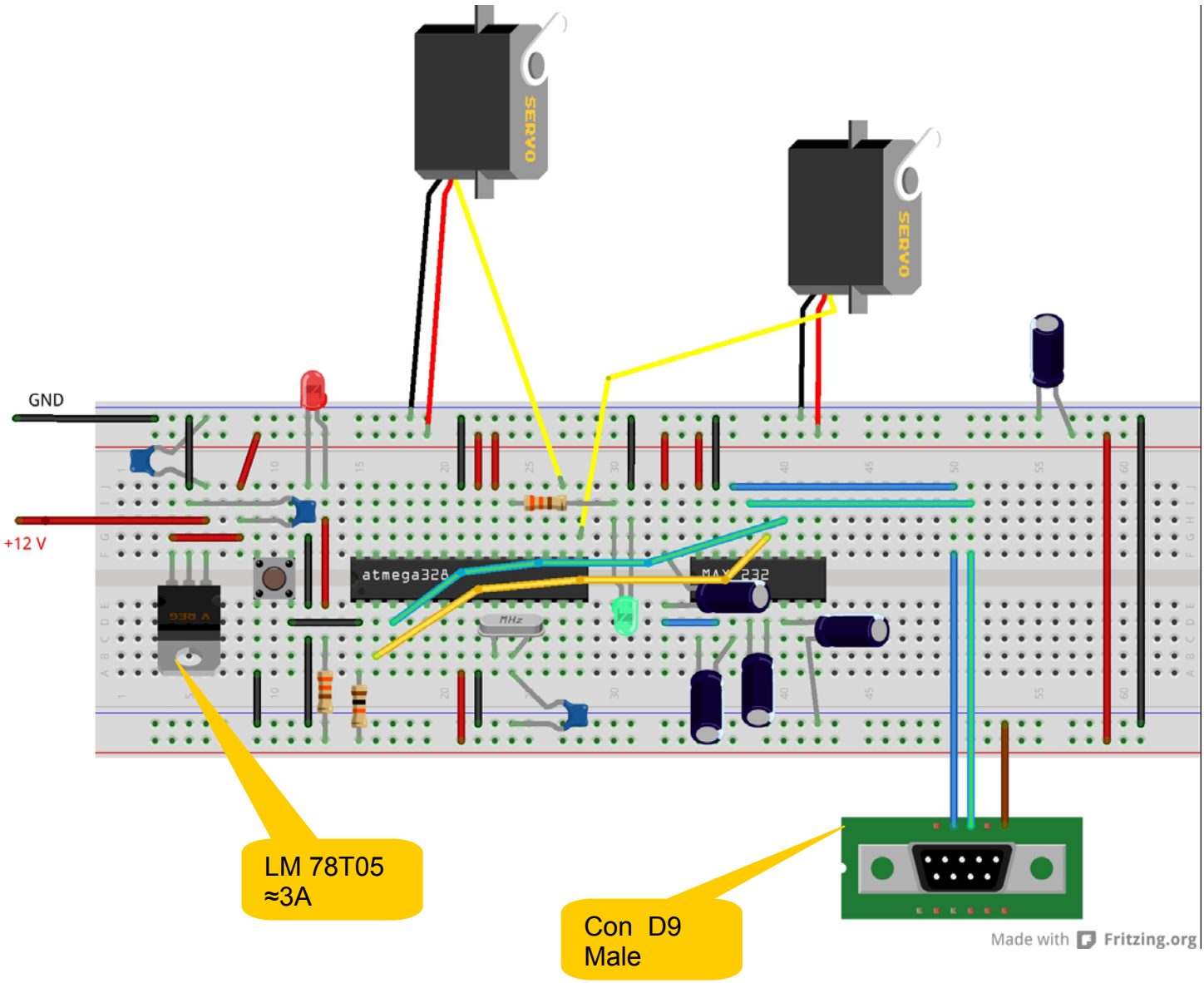
Οι χρόνοι των **ενεργών** παλμών της PWM μπορούν να αλλάξουν, αν τα χαρακτηριστικά του servo μας απαιτούν διαφορετικούς χρόνους παλμών. *attach*.(*pin, min , max*).

Βλέπε *www.arduino.cc* , Reference →Libraries →Servo

Το Arduino Uno στην σειριακή του θύρα , λαμβάνει ή στέλνει ψηφιακά σήματα επιπέδου TTL, μέσα από το UART , (+5 volt , 0) ενώ ο Η/Υ σήματα ± 12 volt ως προς τη γη (πρότυπο σειριακής διασύνδεσης RS-232C). Για το λόγο αυτό συνδέεται στα pins Rx,Tx ένας *μετατροπέας* TTL σε RS-232C, το γνωστό σε όλους IC: MAX 232 , που φαίνεται στο παρακάτω κυκλωματικό διάγραμμα.



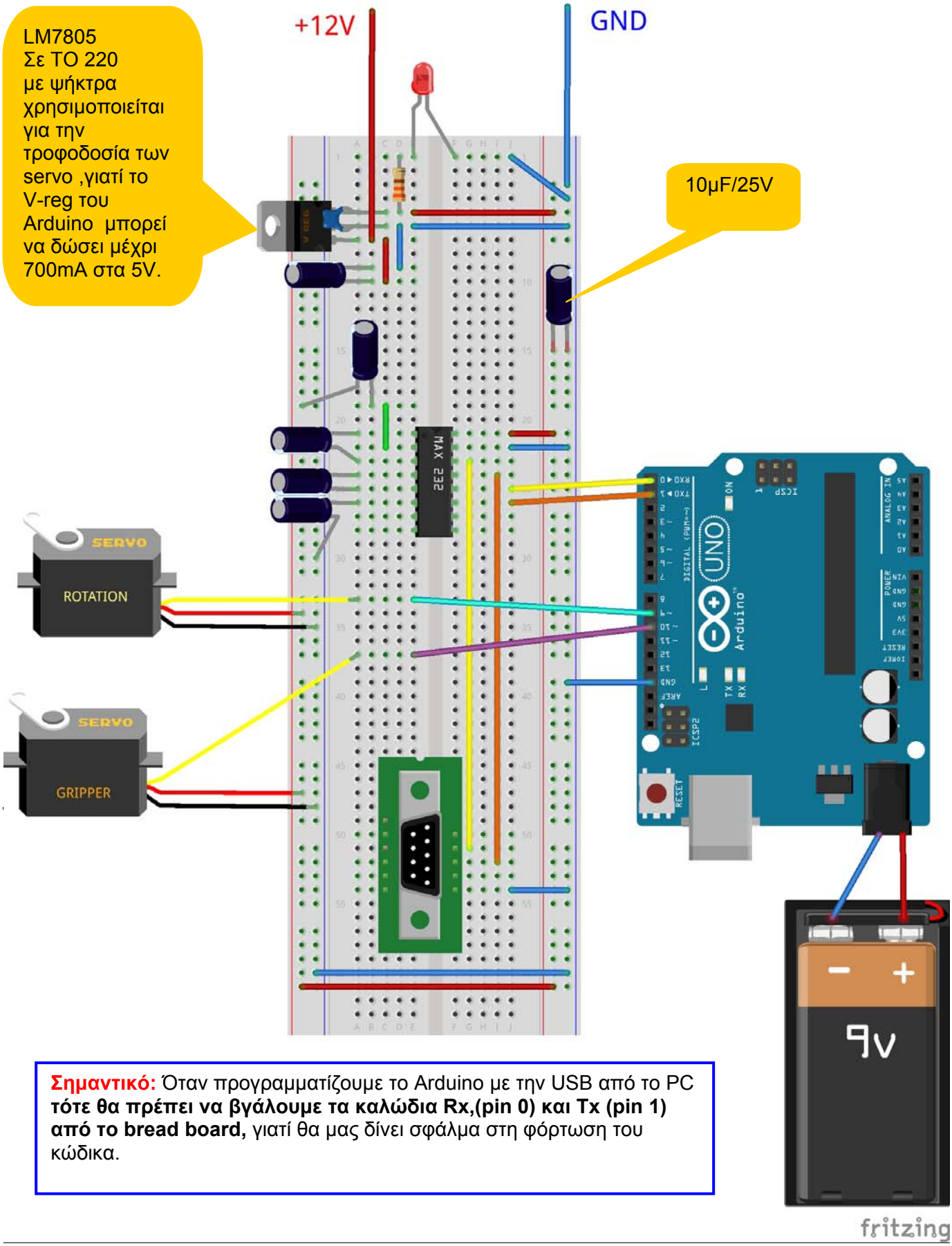
Καλωδίωση στο Bread board (κύκλωμα χωρίς τη χρήση της πλακέτας Arduino Uno)



Συρμάτωση στο bread board ενός Atmega 328p με το MAX 232 , για ασύγχρονη σειριακή επικοινωνία με Η/Υ.

Αν χρησιμοποιηθεί αυτό το κύκλωμα τότε εννοείται ότι ο μ/ε atmega 328p θα προγραμματίζεται στην πλακέτα του Arduino_Uno και θα επανατοποθετείται πάλι στο breadboard.
Συνιστάται η χρήση βάσεων ZIF 28p , και στο breadboard αλλά και στην πλακέτα Arduino UNO για να γίνεται γρήγορα η αντικατάσταση , και να μην κινδυνέψουμε την καταστροφή στα ποδαράκια του IC από τις πολλές δοκιμές.

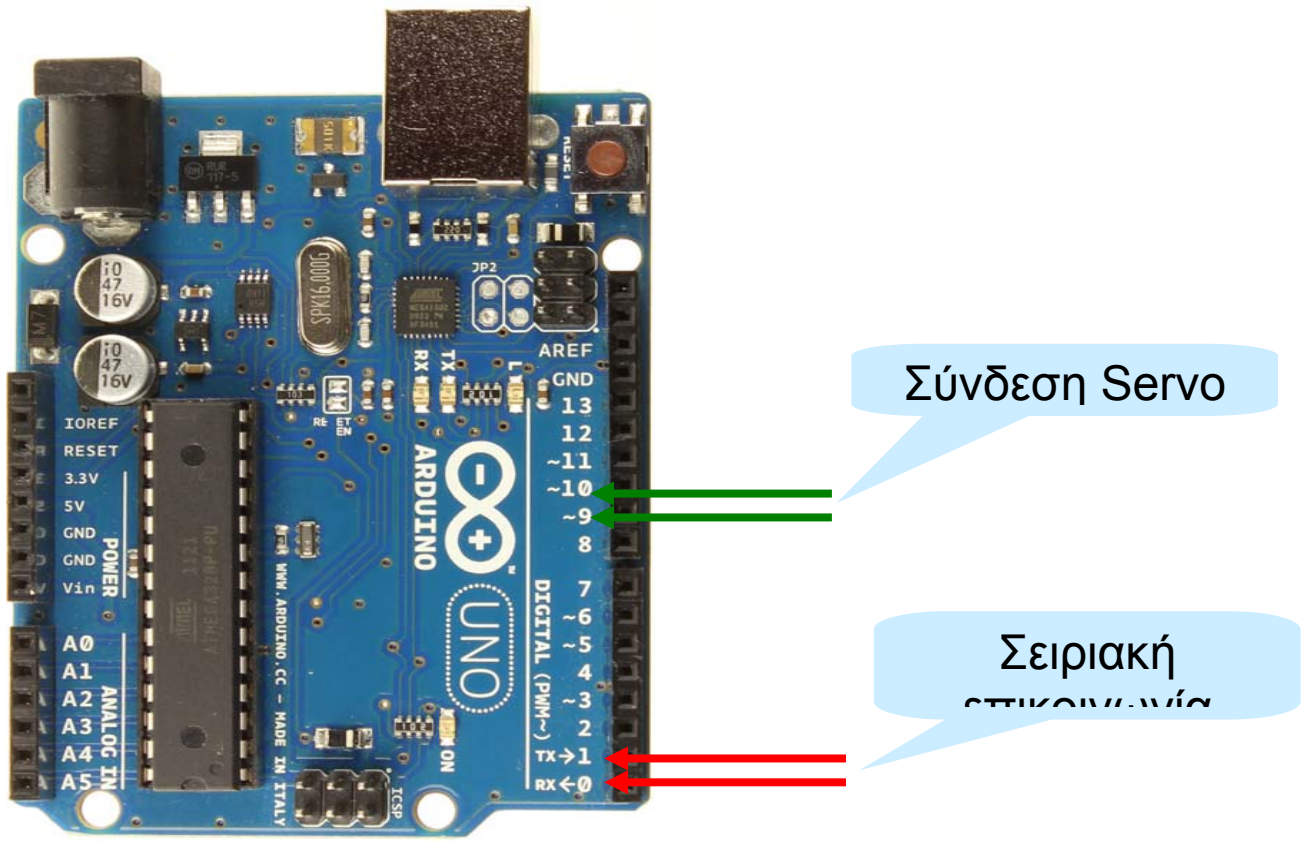
Αν κάνουμε χρήση της πλακέτας Arduino UNO η συρμάτωση φαίνεται παρακάτω.

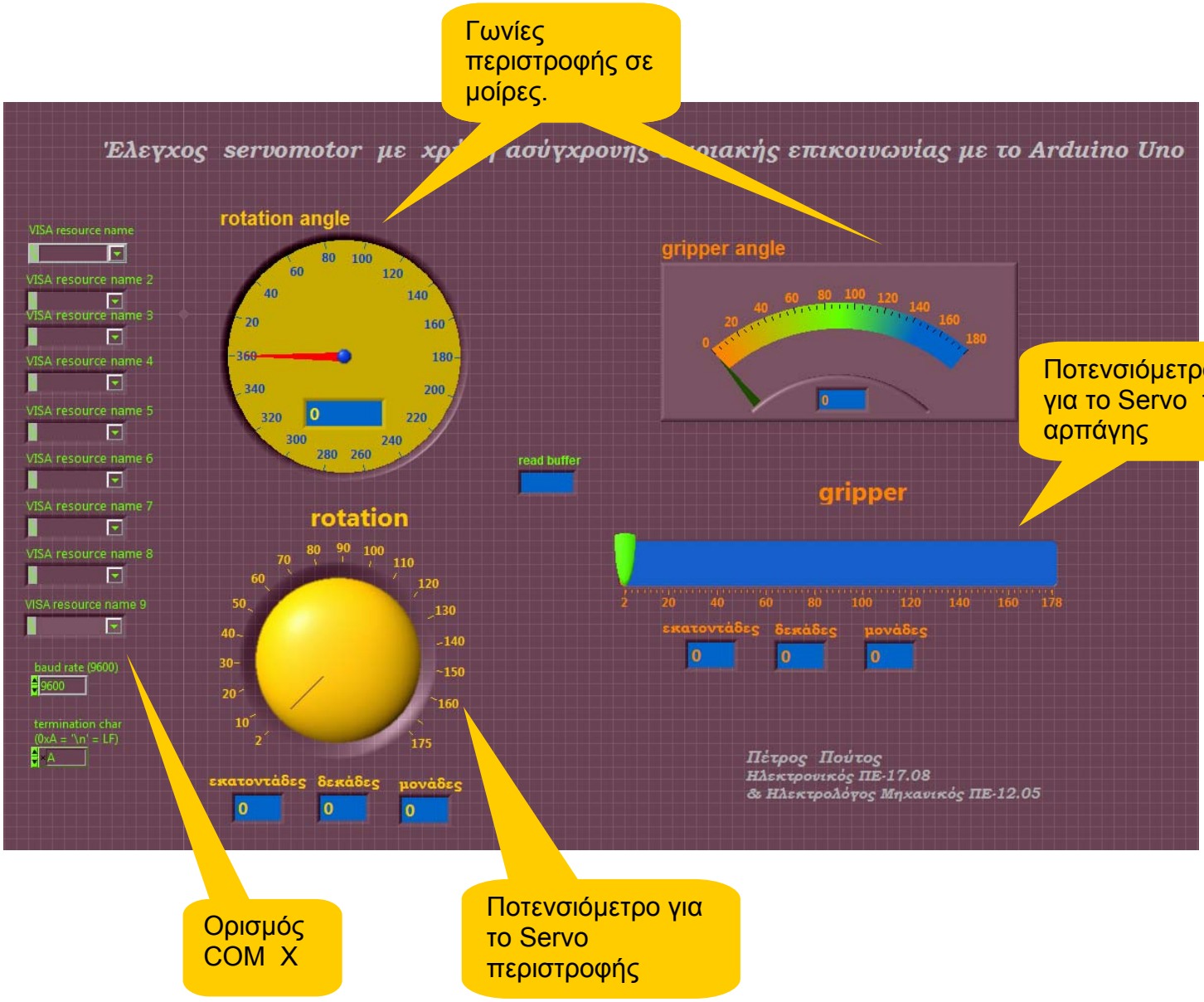


Σημαντικό: Όταν προγραμματίζουμε το Arduino με την USB από το PC **τότε θα πρέπει να βγάλουμε τα καλώδια Rx,(pin 0) και Tx (pin 1) από το bread board,** γιατί θα μας δίνει σφάλμα στη φόρτωση του κώδικα.

Λίστα Εξαρτημάτων

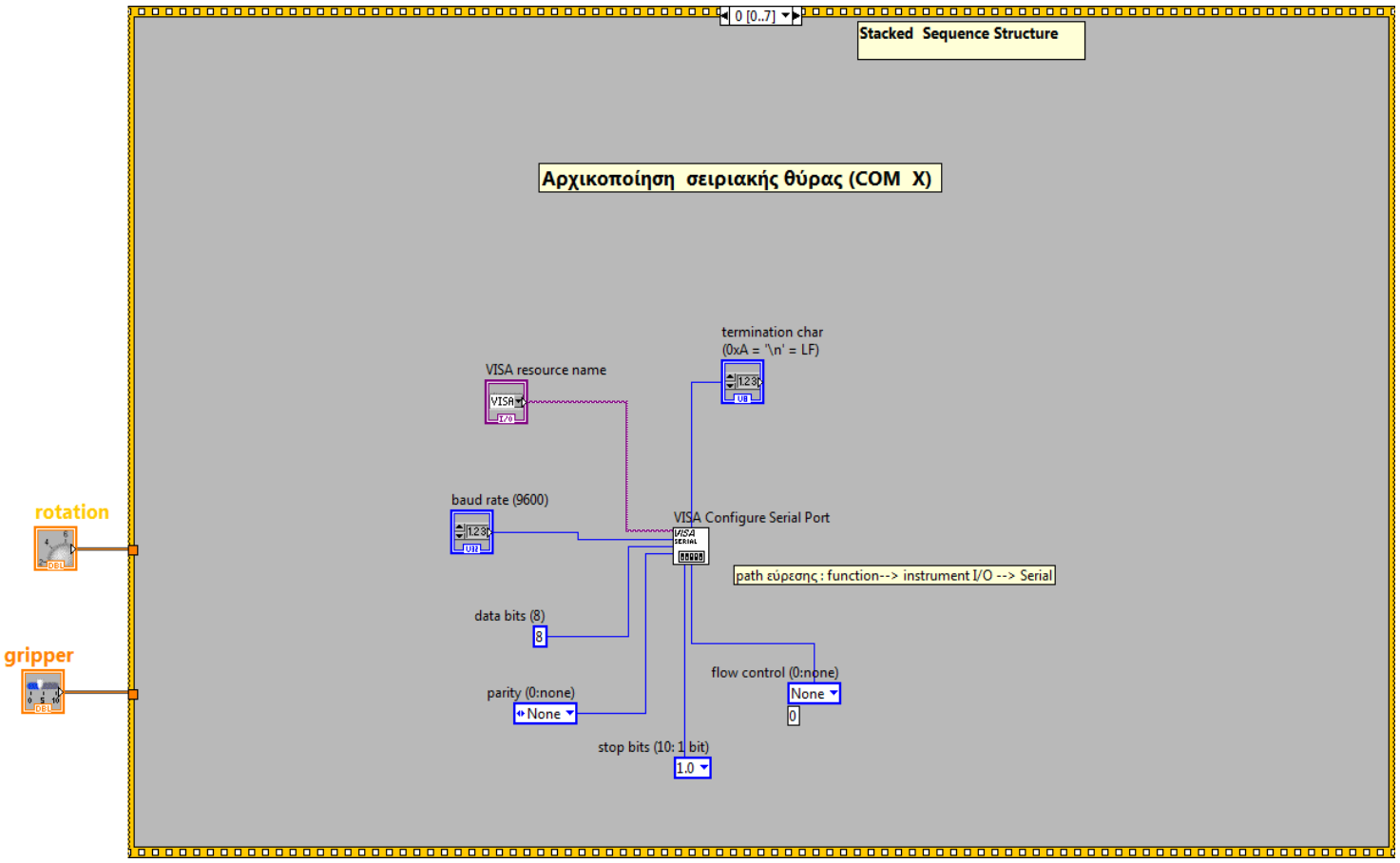
Εξάρτημα	Ποσότητα.
Πλακέτα Arduino Uno	1
Bread board	1
Μπαταρία 9 Volt	1
Μπαταρία 12 volt 2 Ah ή τροφοδοτικό DC	1
Analog Servo motors standard 4,8 – 6 volt	Από 2 έως 6
Βύσμα power male	1
D9 connector	1
Ψήκτρα για TO 220	1
Καλώδια συρμάτωσης	Αρκετά διαφόρων χρωμάτων
Ηλεκτρολυτικός πυκνωτής 10μF/25V	1
Ηλεκτρολυτικός πυκνωτής 1,5μF/25V	5
Πυκνωτής MKT 100nF	1
Αντίσταση 330Ω --1/4W	1
Led 3mm	1
IC LM 7805	1
IC Max 232	1
Καλώδιο σειριακής επικοινωνίας female –female ή καλώδιο μετατροπής USB σε σειριακό	1
Καλώδιο USB για προγραμματισμό Arduino Uno	1
Λογισμικό για το Arduino έκδοση 1.0.1 και άνω	1
Λογισμικό LabView έκδοση 8.5	1

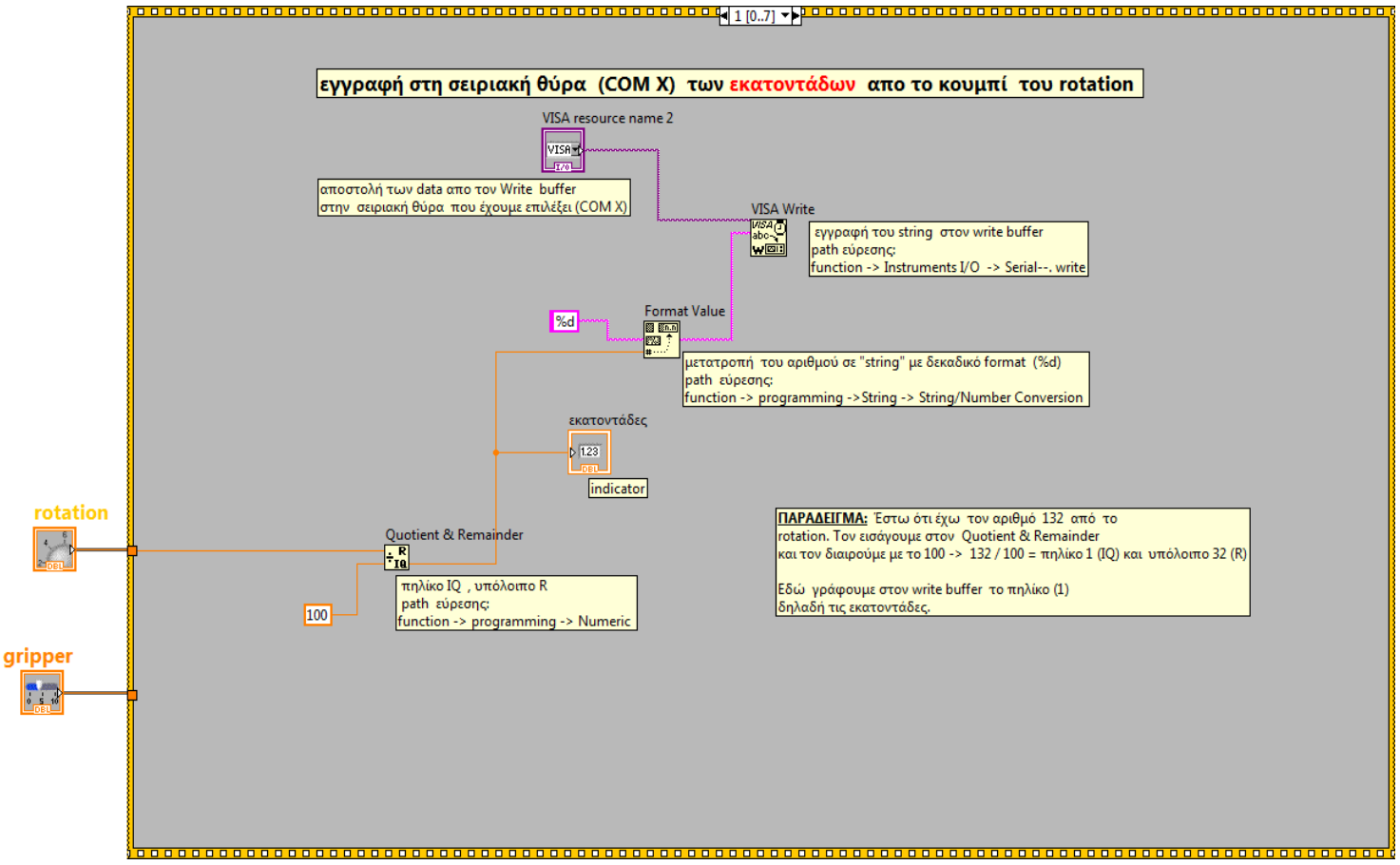


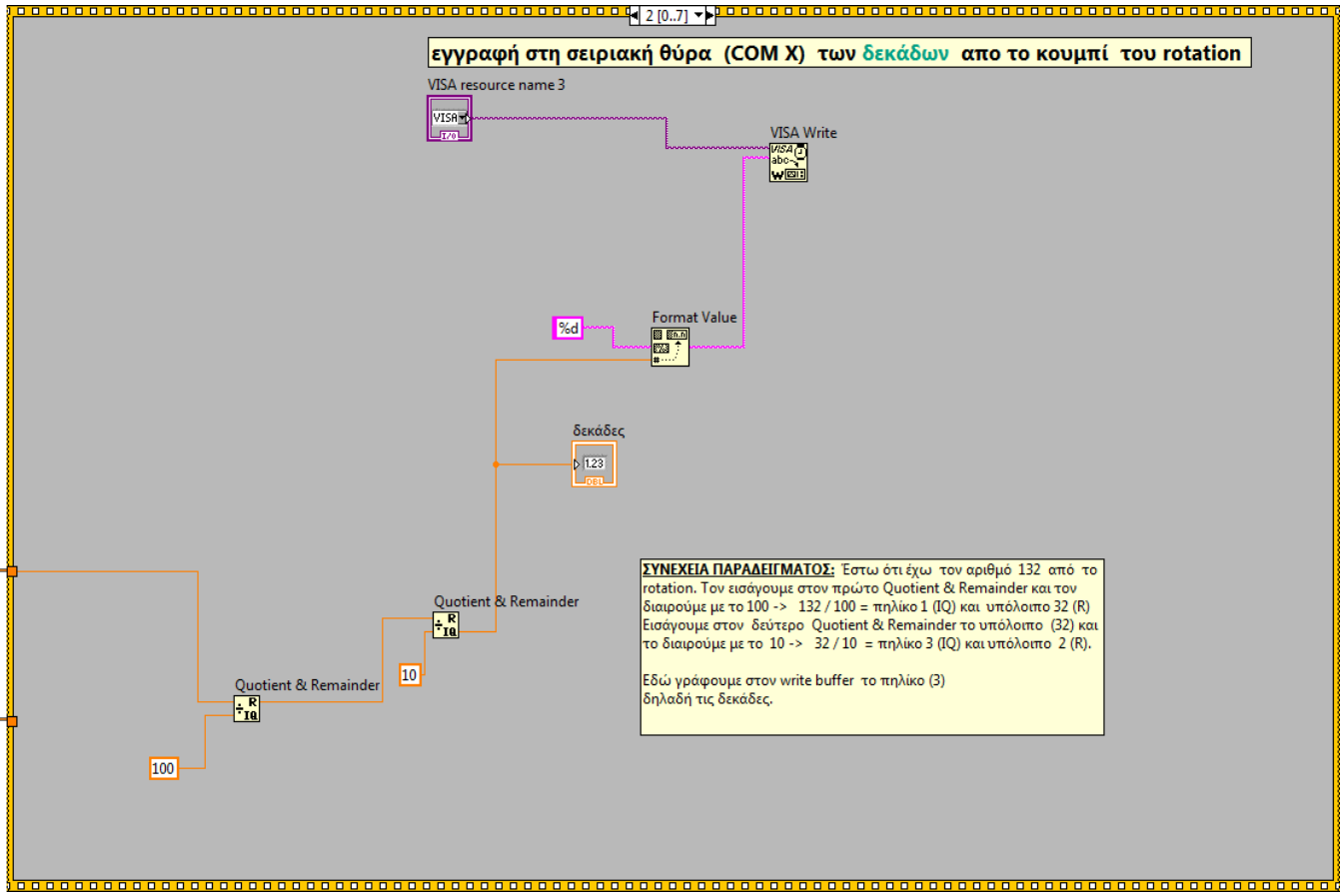


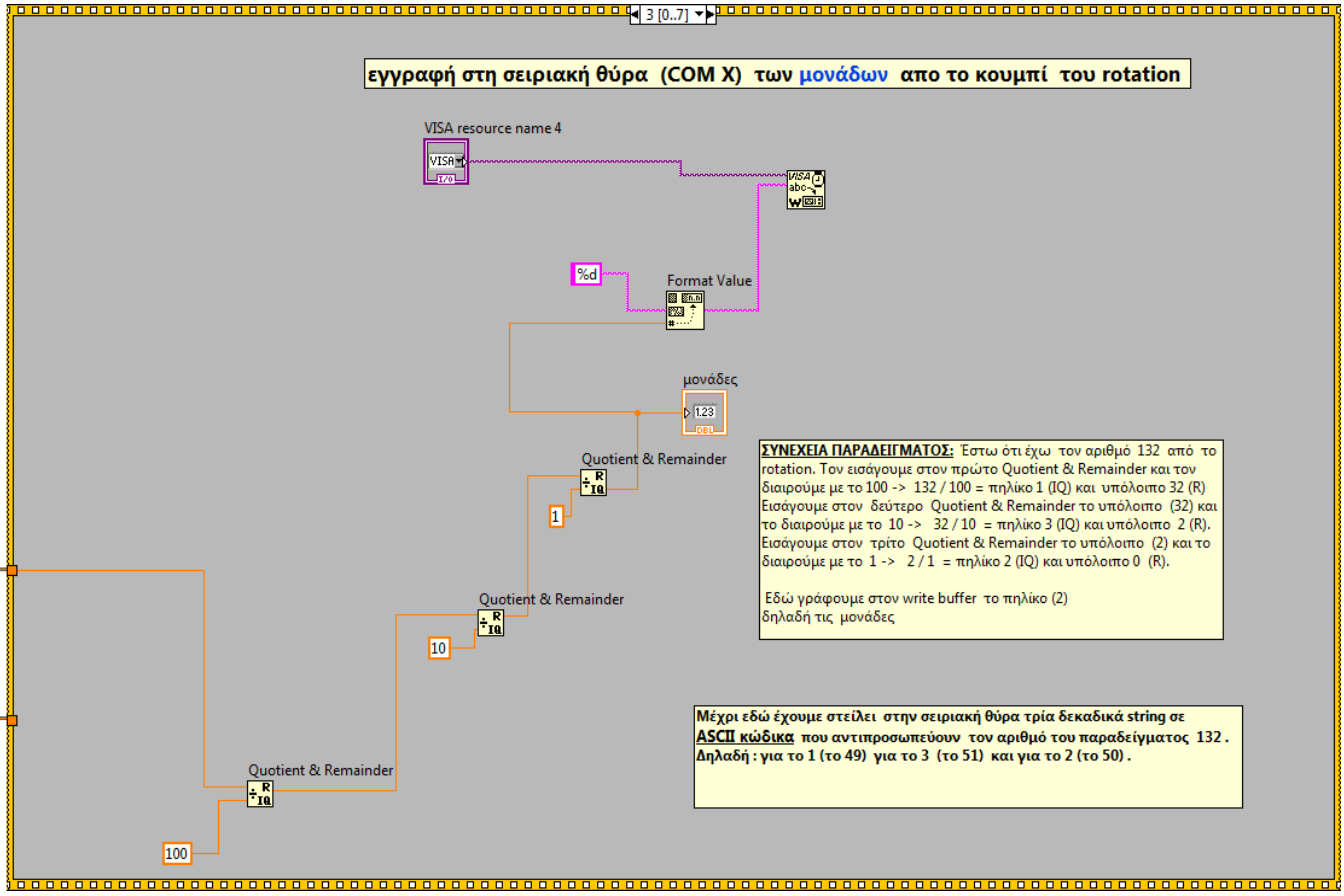
Frond panel LabView

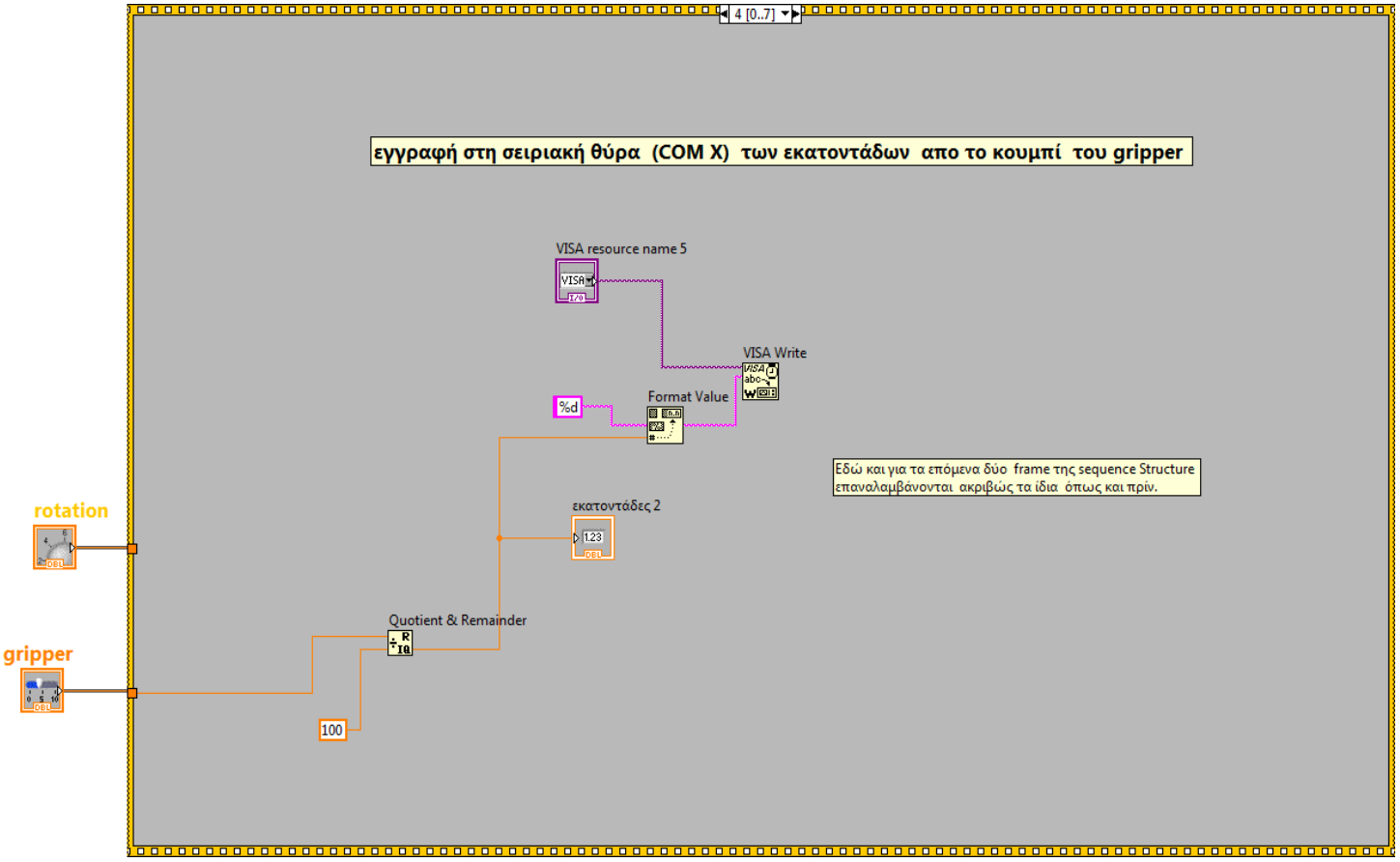
Ο γραφικός κώδικας στο Block diagram

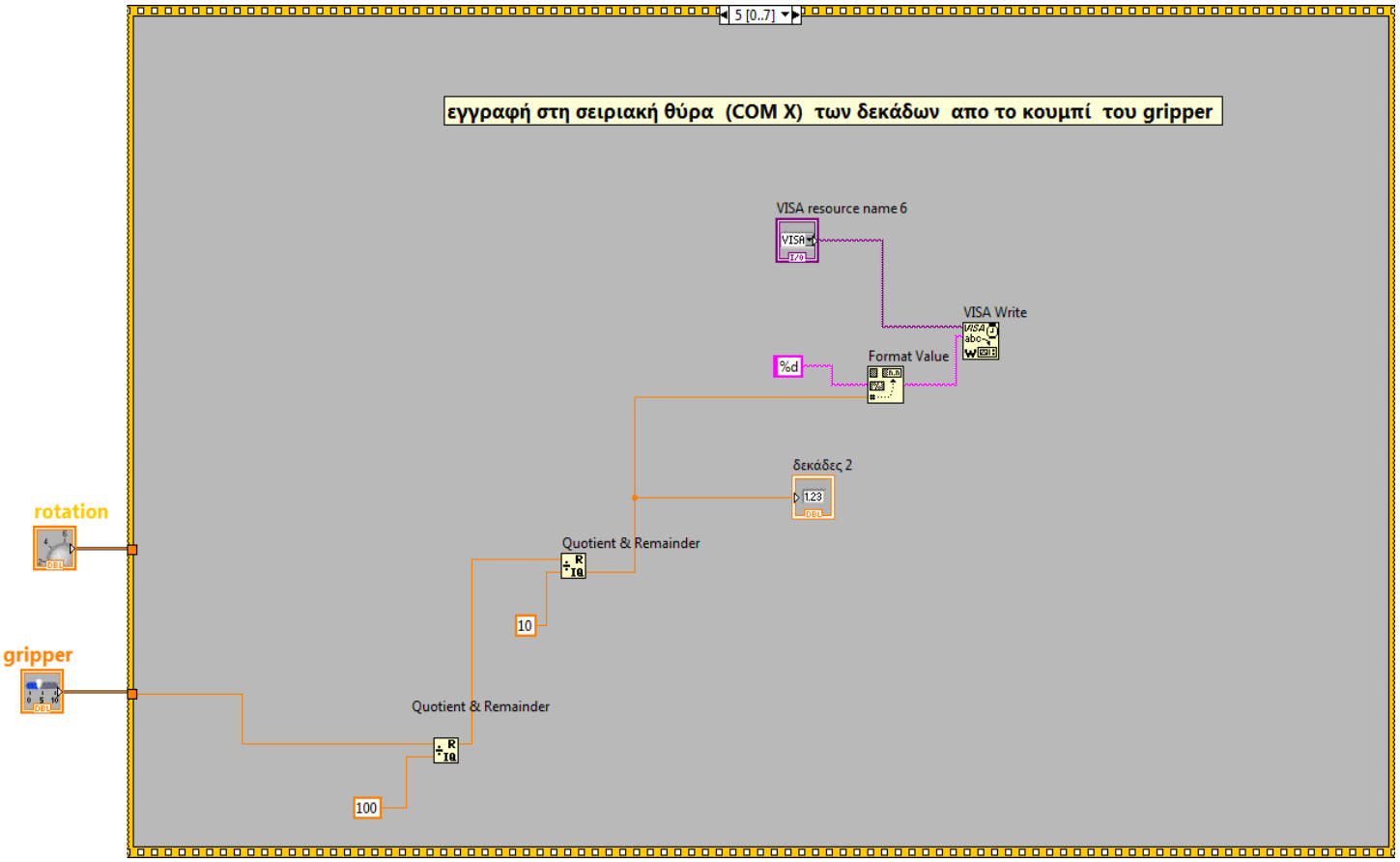


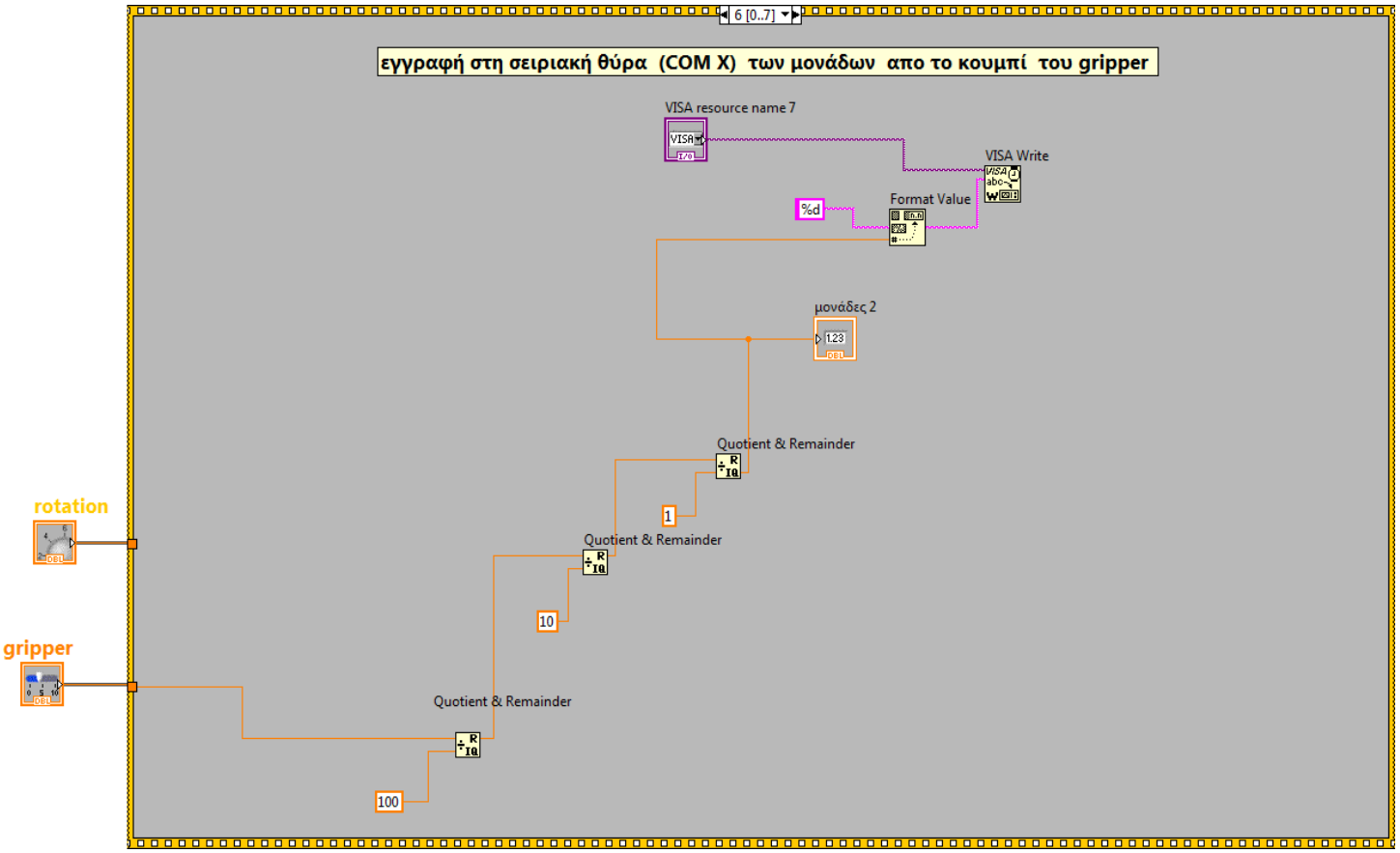


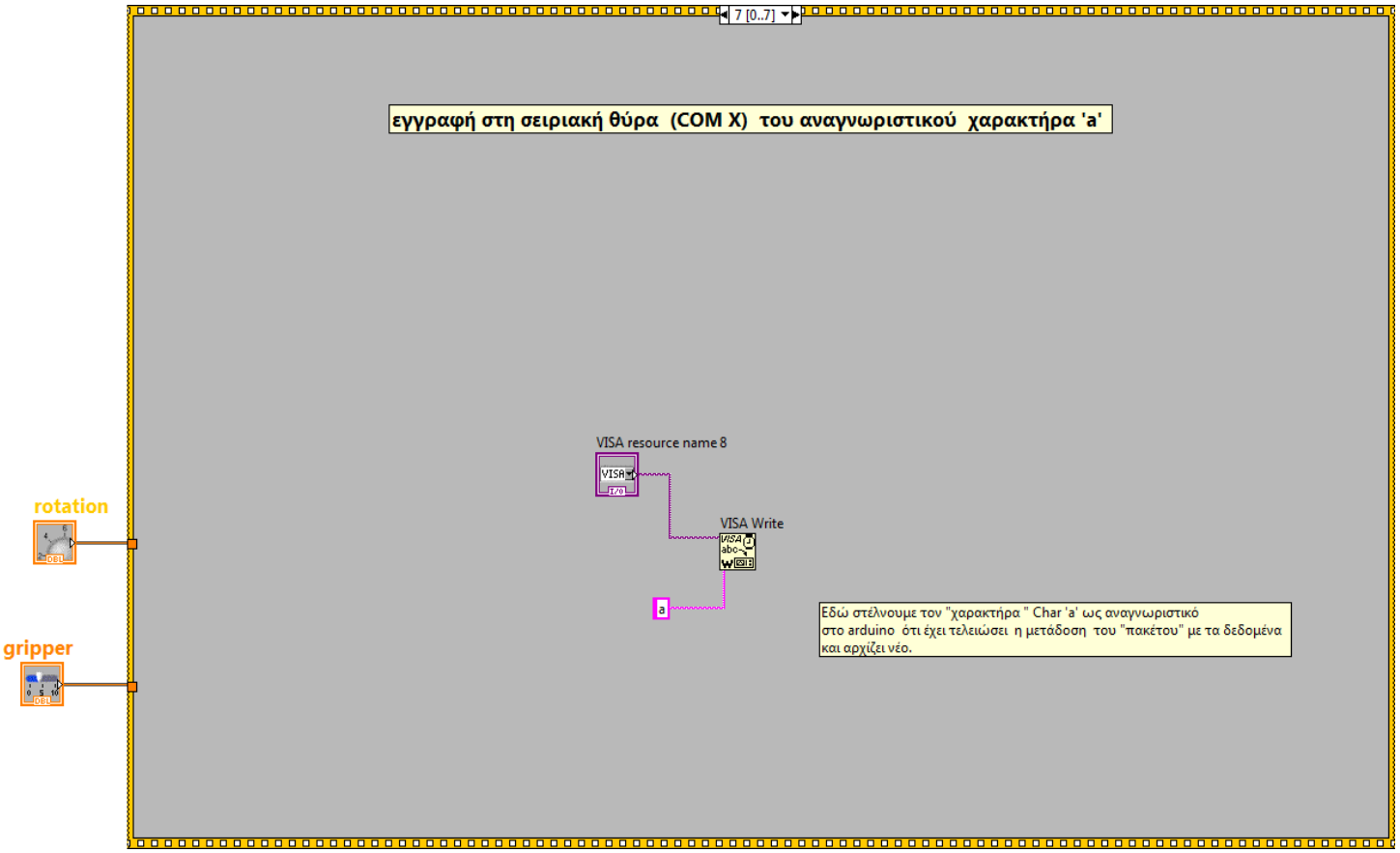

2 [0..7]
εγγραφή στη σειριακή θύρα (COM X) των δεκάδων απο το κουμπί του rotation
VISA resource name 3
VISA Write
%d
Format Value
δεκάδες
1.23
rotation
gripper
Quotient & Remainder
Quotient & Remainder
10
100
ΣΥΝΕΧΕΙΑ ΠΑΡΑΔΕΙΓΜΑΤΟΣ: Έστω ότι έχω τον αριθμό 132 από το rotation. Τον εισάγουμε στον πρώτο Quotient & Remainder και τον διαιρούμε με το 100 -> 132 / 100 = πηλίκο 1 (IQ) και υπόλοιπο 32 (R)
Εισάγουμε στον δεύτερο Quotient & Remainder το υπόλοιπο (32) και το διαιρούμε με το 10 -> 32 / 10 = πηλίκο 3 (IQ) και υπόλοιπο 2 (R).
Εδώ γράφουμε στον write buffer το πηλίκο (3)
δηλαδή τις δεκάδες.


3 [0..7]
εγγραφή στη σειριακή θύρα (COM X) των μονάδων απο το κουμπί του rotation
VISA resource name 4
%d
Format Value
μονάδες
1.23
rotation
gripper
Quotient & Remainder
Quotient & Remainder
Quotient & Remainder
1
10
100
ΣΥΝΕΧΕΙΑ ΠΑΡΑΔΕΙΓΜΑΤΟΣ: Έστω ότι έχω τον αριθμό 132 από το rotation. Τον εισάγουμε στον πρώτο Quotient & Remainder και τον διαιρούμε με το 100 -> 132 / 100 = πηλίκο 1 (IQ) και υπόλοιπο 32 (R)
Εισάγουμε στον δεύτερο Quotient & Remainder το υπόλοιπο (32) και το διαιρούμε με το 10 -> 32 / 10 = πηλίκο 3 (IQ) και υπόλοιπο 2 (R).
Εισάγουμε στον τρίτο Quotient & Remainder το υπόλοιπο (2) και το διαιρούμε με το 1 -> 2 / 1 = πηλίκο 2 (IQ) και υπόλοιπο 0 (R).
Εδώ γράφουμε στον write buffer το πηλίκο (2)
δηλαδή τις μονάδες
Μέχρι εδώ έχουμε στείλει στην σειριακή θύρα τρία δεκαδικά string σε ASCII κώδικα που αντιπροσωπεύουν τον αριθμό του παραδείγματος 132 .
Δηλαδή : για το 1 (το 49) για το 3 (το 51) και για το 2 (το 50) .

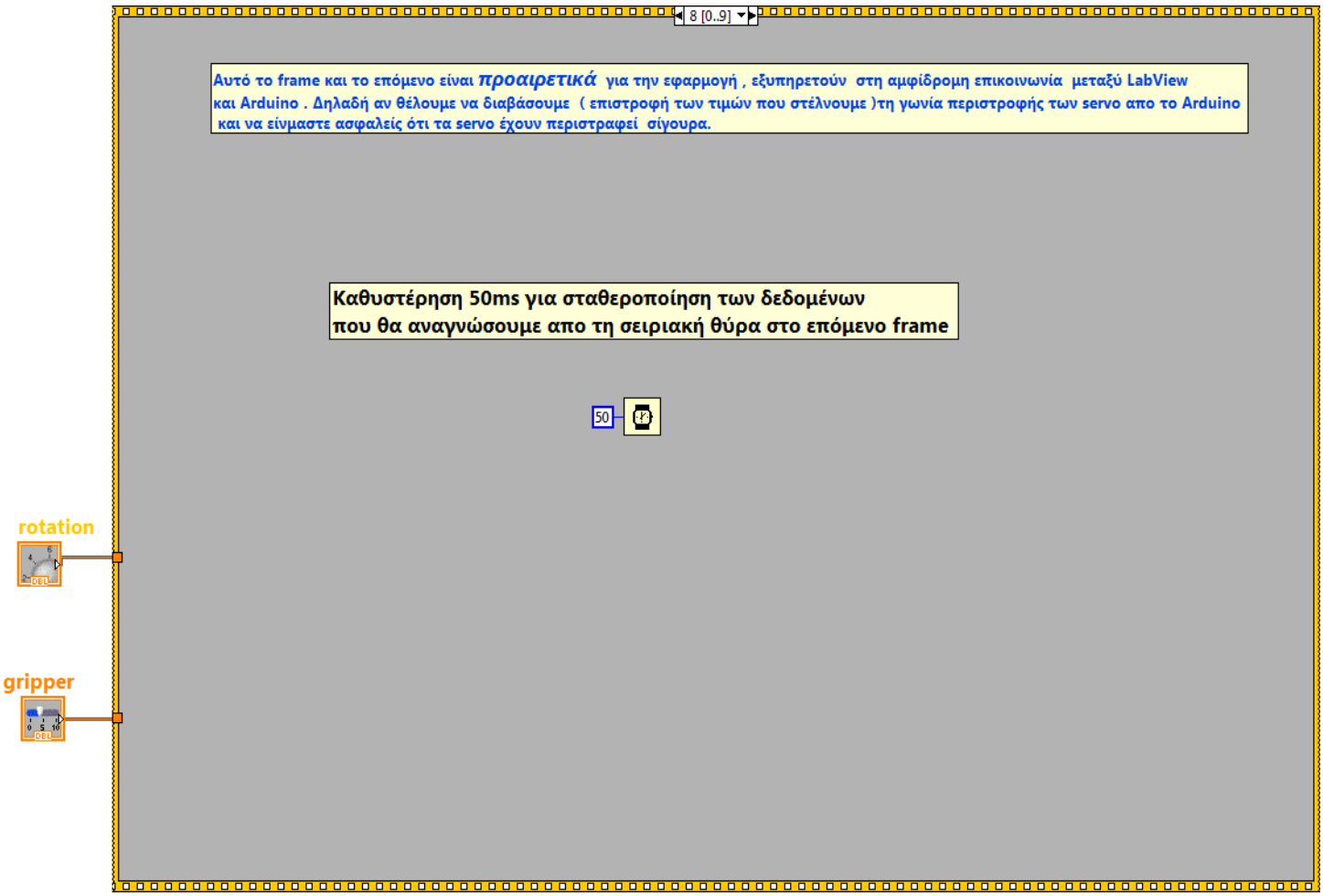

4 [0..7]
εγγραφή στη σειριακή θύρα (COM X) των εκατοντάδων απο το κουμπί του gripper
VISA resource name 5
VISA Write
Format Value
%d
Εδώ και για τα επόμενα δύο frame της sequence Structure επαναλαμβάνονται ακριβώς τα ίδια όπως και πριν.
εκατοντάδες 2
rotation
gripper
Quotient & Remainder
100

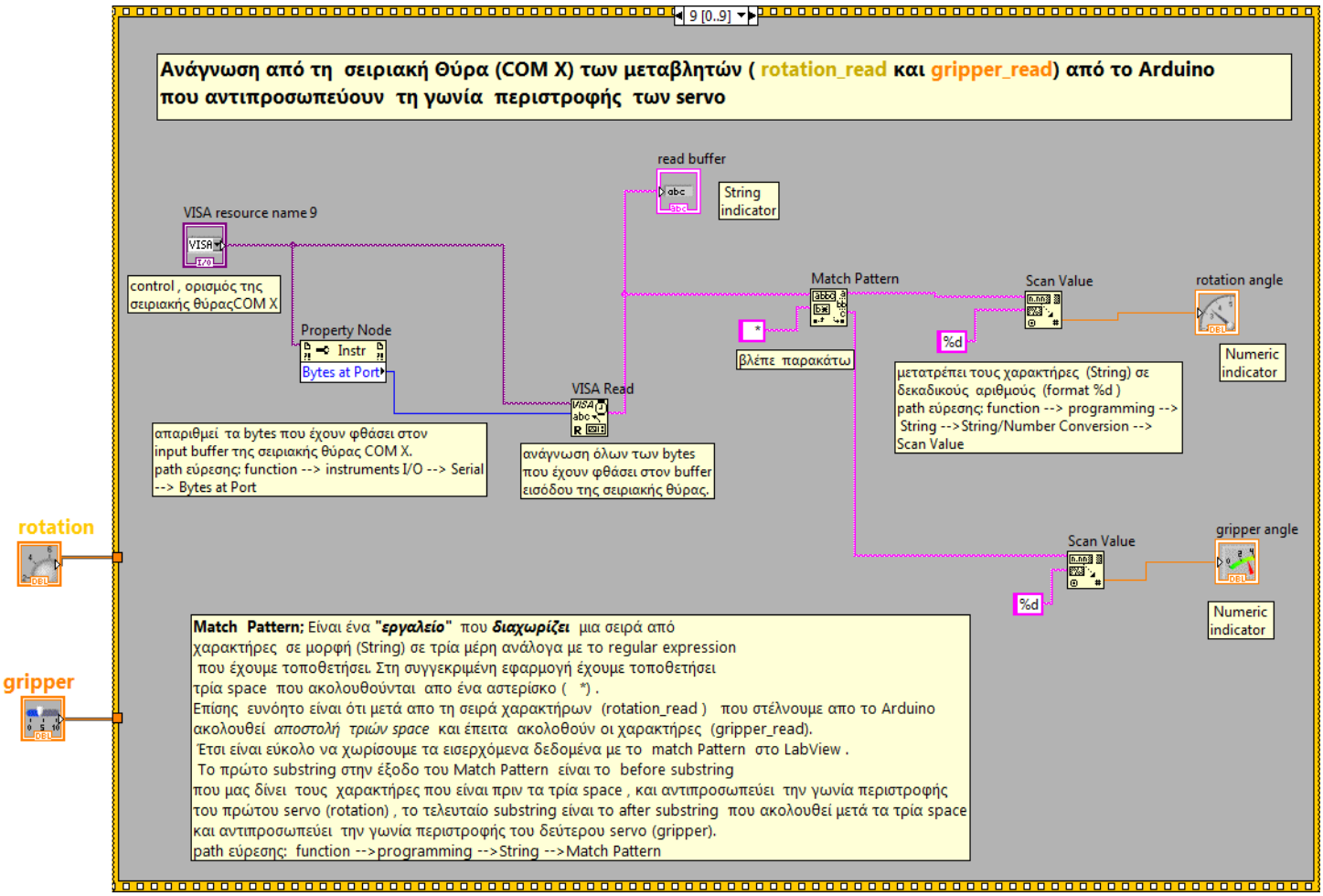

5 [0..7]
εγγραφή στη σειριακή θύρα (COM X) των δεκάδων απο το κουμπί του gripper
VISA resource name 6
VISA Write
Format Value
%d
δεκάδες 2
rotation
gripper
Quotient & Remainder
10
Quotient & Remainder
100





Σημαντικό: *Από τη σειριακή θύρα «φεύγουν» οι χαρακτήρες ASCII* ***κατά σειρά:***
1ος *εκατοντάδες του rotation,* ***2ος*** *δεκάδες του rotation,* ***3ος*** *μονάδες του rotation,* ***4ος*** *εκατοντάδες του gripper,* ***5ος*** *δεκάδες του gripper,* ***6ος*** *μονάδες του gripper,* ***7ος*** *χαρακτήρας 'a'.*


8 [0..9]
Αυτό το frame και το επόμενο είναι προαιρετικά για την εφαρμογή , εξυπηρετούν στη αμφίδρομη επικοινωνία μεταξύ LabView και Arduino . Δηλαδή αν θέλουμε να διαβάσουμε (επιστροφή των τιμών που στέλνουμε)τη γωνία περιστροφής των servo απο το Arduino και να είμαστε ασφαλείς ότι τα servo έχουν περιστραφεί σίγουρα.
Καθυστέρηση 50ms για σταθεροποίηση των δεδομένων που θα αναγνώσουμε απο τη σειριακή θύρα στο επόμενο frame
50
rotation
gripper


9 [0..9]
Ανάγνωση από τη σειριακή Θύρα (COM X) των μεταβλητών (rotation_read και gripper_read) από το Arduino που αντιπροσωπεύουν τη γωνία περιστροφής των servo
read buffer
String indicator
VISA resource name 9
control , ορισμός της σειριακής θύραςCOM X
Property Node
Instr
Bytes at Port
απαριθμεί τα bytes που έχουν φθάσει στον input buffer της σειριακής θύρας COM X.
path εύρεσης: function --> instruments I/O --> Serial --> Bytes at Port
VISA Read
ανάγνωση όλων των bytes που έχουν φθάσει στον buffer εισόδου της σειριακής θύρας.
Match Pattern
βλέπε παρακάτω
Scan Value
%d
μετατρέπει τους χαρακτήρες (String) σε δεκαδικούς αριθμούς (format %d)
path εύρεσης: function --> programming --> String -->String/Number Conversion --> Scan Value
rotation angle
Numeric indicator
Scan Value
%d
gripper angle
Numeric indicator
Match Pattern: Είναι ένα "εργαλείο" που διαχωρίζει μια σειρά από χαρακτήρες σε μορφή (String) σε τρία μέρη ανάλογα με το regular expression που έχουμε τοποθετήσει. Στη συγγεκριμένη εφαρμογή έχουμε τοποθετήσει τρία space που ακολουθούνται απο ένα αστερίσκο (*) .
Επίσης ευνόητο είναι ότι μετά απο τη σειρά χαρακτήρων (rotation_read) που στέλνουμε απο το Arduino ακολουθεί αποστολή τριών space και έπειτα ακολοθούν οι χαρακτήρες (gripper_read).
Έτσι είναι εύκολο να χωρίσουμε τα εισερχόμενα δεδομένα με το match Pattern στο LabView .
Το πρώτο substring στην έξοδο του Match Pattern είναι το before substring που μας δίνει τους χαρακτήρες που είναι πριν τα τρία space , και αντιπροσωπεύει την γωνία περιστροφής του πρώτου servo (rotation) , το τελευταίο substring είναι το after substring που ακολουθεί μετά τα τρία space και αντιπροσωπεύει την γωνία περιστροφής του δεύτερου servo (gripper).
path εύρεσης: function -->programming -->String -->Match Pattern
rotation
gripper

Κώδικας στο Arduino με αναλυτικά σχόλια.

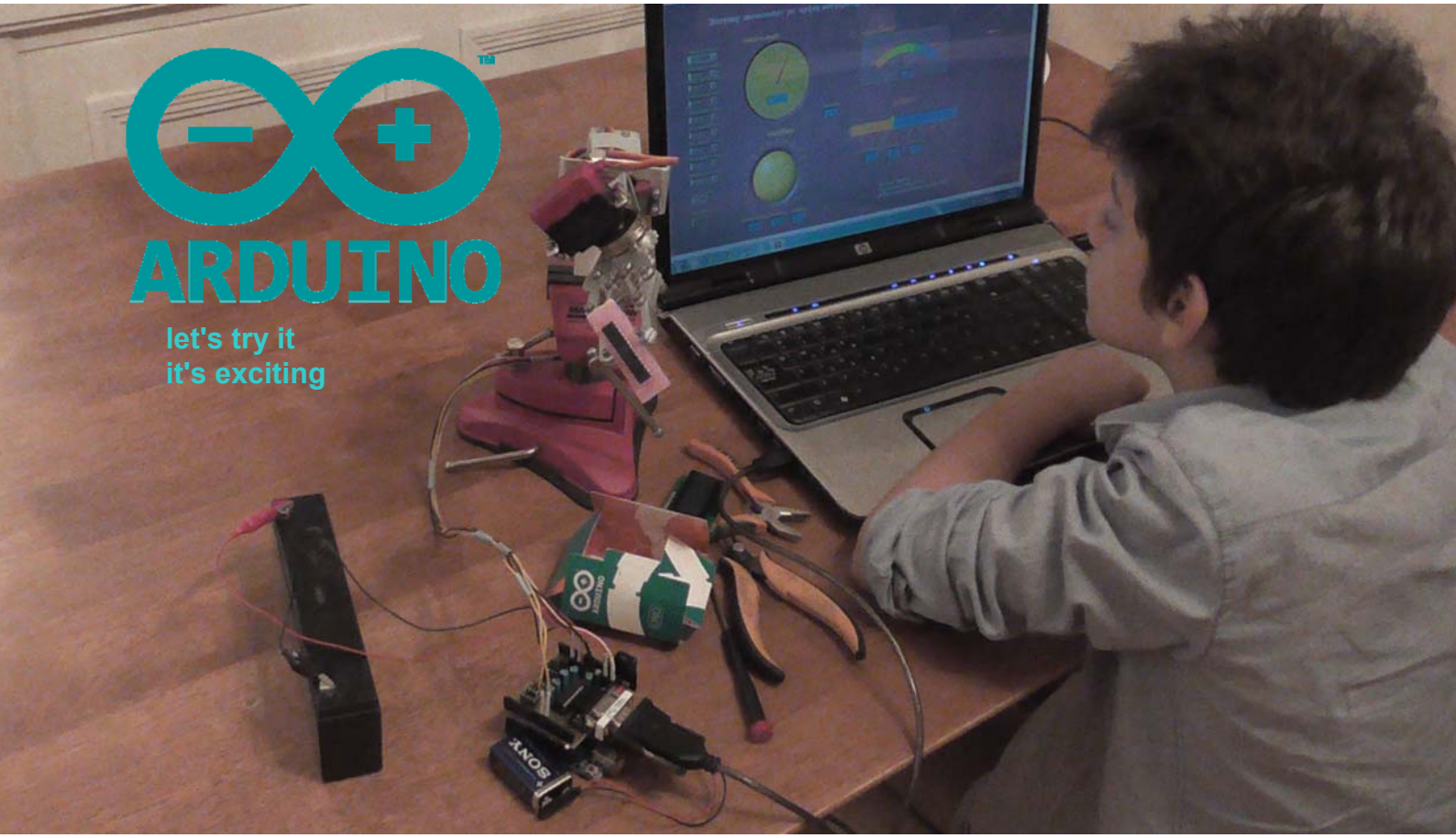
```
#include <Servo.h>  // εισαγωγή standard  βιβλιοθήκης Servo για έλεγχο σερβοκινητήρων

//Define variables  ορισμός μεταβλητών που θα χρησιμοποιηθούν στο πρόγραμμα.

// output servo pins  // ορίζουμε τα ποδαράκια στην πλακέτα του Arduino Uno που θα
// συνδέσουμε το καλώδιο του servo που θα δεχτεί την PWM τάση. (πορτοκαλί ή άσπρο).
const int servo1 = 9;  // εδώ συνδέουμε τα καλώδια των servomotors στα ποδαράκια 9 και 10
const int servo2 = 10;

//create servo object to control a servo
// ορίζουμε δύο «αντικείμενα» myservo1 και myservo2 με το χαρακτηριστικό Servo κατά
απαίτηση της βιβλιοθήκης Servo.h (είναι σαν μεταβλητές με ειδικό προσδιορισμό )
Servo myservo1;
Servo myservo2;

int rotation = 0; // ορισμός και αρχικοποίηση της ακεραίας μεταβλητής rotation  που
                  //αντιπροσωπεύει τις μοίρες περιστροφής του άξονα του πρώτου servo
int rot_h = 0; // ορισμός και αρχικοποίηση της ακεραίας μεταβλητής rot_h
               //  ( από το rotation _hundreds ) που αντιπροσωπεύει τις εκατοντάδες
int rot_d = 0; // το ίδιο για τις δεκάδες
int rot_u = 0; // το ίδιο για τις μονάδες

int gripper =0; // ακριβώς τα ίδια με παραπάνω για το δεύτερο servo.
int grip_h =0;
int grip_d =0;
int grip_u =0;

int rotation_read = 0; // ορισμός και αρχικοποίηση των μεταβλητών που θα χρησιμοποιηθούν
                       //  για ανάγνωση της γωνίας περιστροφής του άξονα του servo και θα
                         σταλούν στο LabView
int gripper_read = 0;

char incomingChar; // ορισμός της μεταβλητής incomingChar με χαρακτηρισμό char
                   // (χαρακτήρας) θα χρησιμοποιηθεί για τον «αναγνωριστικό» 'a'
                     //  που στέλνεται από το LabView

void setup() {                // αυτός ο βρόχος void  setup() {.....} εκτελείται μία φορά και
                              // συνήθως χρησιμοποιείται για αρχικοποιήσεις .

 // Servo
  myservo1.attach(servo1); // εντολή της βιβλιοθήκης Servo.h η οποία αντιστοιχεί τις
                           // «οντότητες» myservo στα αντίστοιχα pins  του Arduino
  myservo2.attach(servo2);

//initialize Serial
  Serial.begin(9600);  // αρχικοποίηση της σειριακής θύρας με «ρυθμό» 9600 bps

  delay(500); //  καθυστέρηση 500msec για να σταθεροποιηθεί η αρχικοποίηση της σειριακής

} // κλείσιμο του βρόχου void  setup()
```

```
// Ο παρακάτω βρόχος  Void loop() {……} εκτελείται συνεχώς


void loop() {

  while(Serial.available()>7){    // βρόχος while () υπό συνθήκη ,αν στον buffer  εισόδου
                                 //  της σειριακής έχουν φθάσει περισσότερα από 7 bytes
                                 //τότε εκτελείται. Τα 7 bytes είναι: τρία από τις
                                 //εκατοντάδες ,δεκάδες, μονάδες του rotation , τρία από τα
                                 //αντίστοιχα του gripper και ο αναγνωριστικός  χαρακτήρας‘a’
  incomingChar = Serial.read();  // διαβάζουμε την σειριακή θύρα ( «ψάχνουμε» τον ‘a’)

 if     (incomingChar == 'a'){       // βρόχος if : αν έχει έρθει ο ’a’ τότε εκτελείται.

       rot_h =  Serial.read()-'0'; // διαβάζουμε  από τη σειριακή θύρα με τη σειρά τους
                                   //  χαρακτήρες ASCII που έρχονται από το LabView
                                   //και ταυτόχρονα αφαιρούμε το ‘0’  για να κάνουμε
                                    // τους χαρακτήρες αριθμητικές ποσότητες.

        rot_d =  Serial.read()-'0';
        rot_u =  Serial.read()-'0';
        grip_h = Serial.read()-'0';
        grip_d = Serial.read()-'0';
        grip_u = Serial.read()-'0';

   rotation = (rot_h*100)+(rot_d*10)+(rot_u*1); // εδώ δίνουμε στους αντίστοιχους αριθμούς
                                                //  δεκαδικά βάρη  ώστε να αντιπροσωπεύουν
                                               // τις μοίρες περιστροφής του άξονα του servo
   rotation = constrain(rotation, 2, 175);          // περιορισμός της τιμής rotation μεταξύ
                                                     // (2°,175°) για προστασία των γραναζιών
                                                        //του servo
   gripper = (grip_h*100)+(grip_d*10)+(grip_u*1);
   gripper = constrain(gripper, 2, 178);

   myservo1.write(rotation);       // εδώ γράφουμε την τιμή των μοιρών περιστροφής, στην
                                   // αντίστοιχη  «οντότητα» για να παραχθεί από τη βιβλιοθήκη
                                   // Servo.h  η κατάλληλη PWM που θα περιστρέψει τα servo
   myservo2.write(gripper);

    rotation_read = myservo1.read(); // ανάγνωση του αριθμού των μοιρών περιστροφής του
                                     //  κάθε σερβοκινητήρα.
    gripper_read = myservo2.read();

  Serial.print(rotation_read); // εγγραφή της παραπάνω τιμής στην σειριακή θύρα,
                               // (εννοείται σε χαρακτήρες ASCII) για να σταλεί στο LabView

   Serial.print("   ");        // αναγνωριστικό διαχωρισμού μεταξύ των δύο τιμών(τρία space)

   Serial.print(gripper_read);

   Serial.flush(); // καθαρισμός του buffer της σειριακής.

   }  // κλείσιμο του βρόχου if ()

  }    // κλείσιμο του βρόχου While ()

}     // κλείσιμο του βρόχου void loop()
```

Κώδικας στο Arduino χωρίς σχόλια.

```
#include <Servo.h>

//Define variables

// output servo pins
const int servo1 = 9;
const int servo2 = 10;

//create servo object to control a servo
Servo myservo1;
Servo myservo2;

int rotation = 0;
int rot_h = 0;
int rot_d = 0;
int rot_u = 0;

int gripper =0;
int grip_h =0;
int grip_d =0;
int grip_u =0;

int rotation_read = 0;
int gripper_read = 0;

char incomingChar;

void setup()

 // Servo
  myservo1.attach(servo1);
  myservo2.attach(servo2);

//initialize Serial
  Serial.begin(9600);
  delay(500);

}


void loop() {

  while(Serial.available()>7){
  incomingChar = Serial.read();
 if    (incomingChar == 'a'){
       rot_h =  Serial.read()-'0';

        rot_d =  Serial.read()-'0';
        rot_u =  Serial.read()-'0';
        grip_h = Serial.read()-'0';
        grip_d = Serial.read()-'0';
        grip_u = Serial.read()-'0';
```

```
    rotation = (rot_h*100)+(rot_d*10)+(rot_u*1);
    rotation = constrain(rotation, 2, 175);
    gripper = (grip_h*100)+(grip_d*10)+(grip_u*1);
    gripper = constrain(gripper, 2, 178);

    myservo1.write(rotation);
    myservo2.write(gripper);

      rotation_read = myservo1.read();
      gripper_read = myservo2.read();

     Serial.print(rotation_read);

     Serial.print("    ");

     Serial.print(gripper_read);

     Serial.flush();

     }

  }

}
```



6η Εφαρμογή: Έλεγχος LED Matrix με Arduino

Χρήση του αναπτυξιακού συστήματος Arduino Uno για τον έλεγχο ενός LED Matrix 6X6

Π. Μπαλούρδος

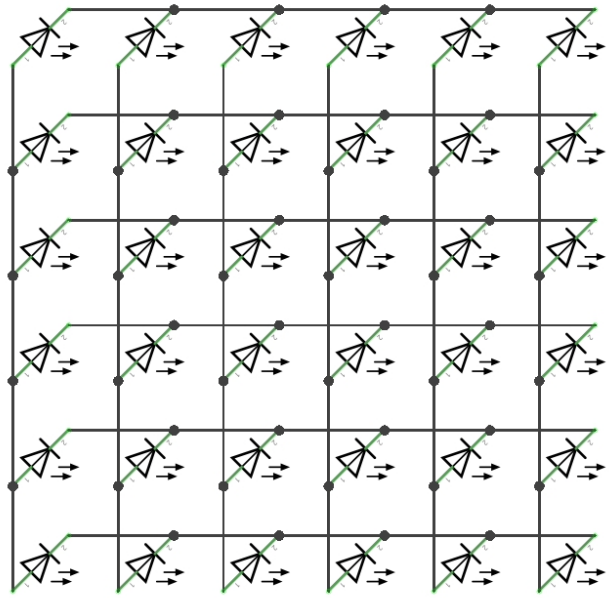
Υλικά

Τα υλικά που χρειάστηκαν για αυτή την εφαρμογή, είναι:

- Ένα Arduino Uno
- 36 LED
- 12 Αντιστάσεις
- 6 Τρανζίστορ
- USB καλώδιο

LED Matrix

Ένα LED Matrix αποτελείται από μεγάλο αριθμό LED και επομένως δεν είναι εφικτή η ανεξάρτητη σύνδεση κάθε LED με το κύκλωμα ελέγχου. Για να μειώσουμε τις συνδέσεις ενός LED Matrix με το κύκλωμα ελέγχου, χρησιμοποιούμε μια μέθοδο πολυπλεξίας η οποία στηρίζεται στο ότι η εικόνα παραμένει στο αισθητήριο της όρασης μετά τον εξωτερικό ερεθισμό (Μετείκασμα – Persistence of Vision). Τα LED του LED Matrix συνδέονται όπως στο Σχήμα 1.



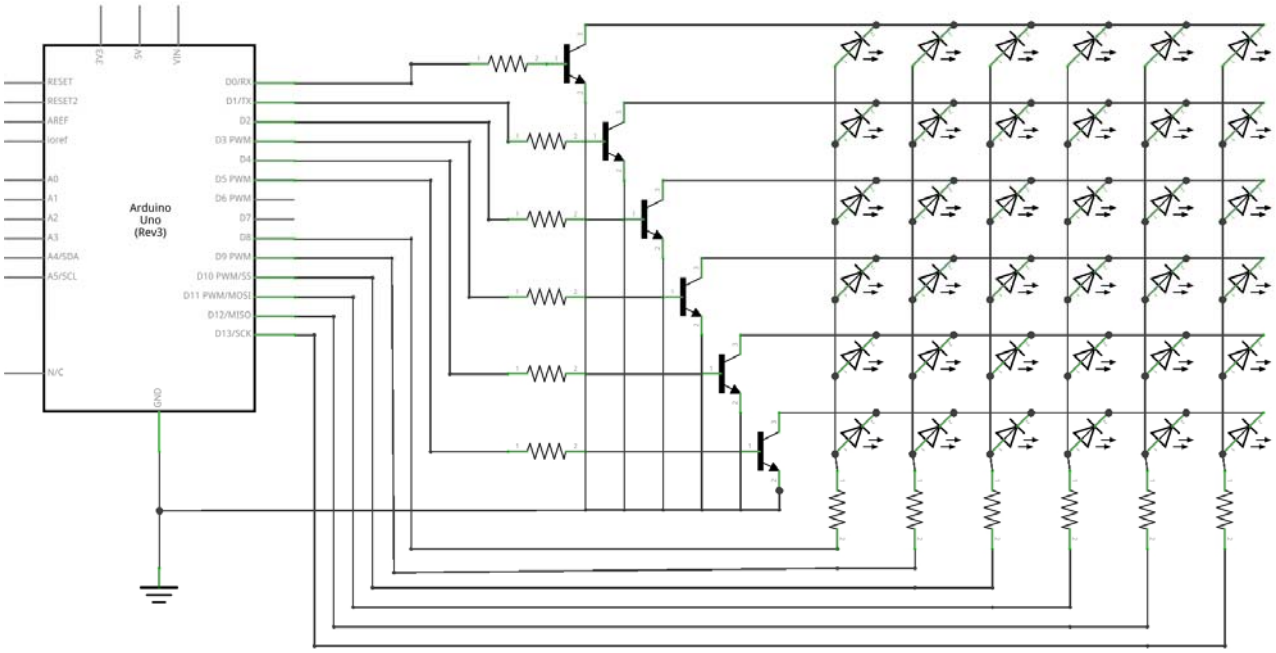
Σχήμα 1

Οι κάθοδοι των LED συνδέονται μεταξύ τους ανά γραμμή, ενώ οι άνοδοι των LED συνδέονται μεταξύ τους ανά στήλη. Η μέθοδος βασίζεται στο ότι κάθε δεδομένη χρονική στιγμή μόνο μια γραμμή του Matrix είναι αναμμένη. Οι γραμμές ανάβουν διαδοχικά αρκετά γρήγορα, ώστε το μάτι μας έχει την αίσθηση ότι τα LED είναι σταθερά αναμένα. Συνεπώς για τον έλεγχο του Matrix του σχήματος 1 χρειαζόμαστε 6 καλώδια για τον έλεγχο των γραμμών και 6 καλώδια για τον έλεγχο των στηλών, ώστε για κάθε γραμμή να μπορούμε να επιλέξουμε ποια LED πρέπει να είναι αναμμένα.

Συνδεσμολογία με το Arduino

Οι στήλες του Matrix (άνοδοι) συνδέονται στις ψηφιακές θύρες 8 έως 13 του Arduino, μέσω αντιστάσεων, ενώ οι γραμμές του Matrix (κάθοδοι) συνδέονται στις

ψηφιακές θύρες 0 έως 5, μέσω απλών ηλεκτρονικών διακοπτών με τρανζίστορ (Σχήμα 2).



Σχήμα 2

Πρόγραμμα

```
// Το πρόγραμμα αυτό ελέγχει ένα LED Matrix 6X6 με τη μέθοδο της πολυπλεξίας.
// Ορισμός θυρών: 0-5 είναι οι γραμμές του matrix (από πάνω προς τα κάτω)
// και 8-13 είναι οι στήλες (από αριστερά προς τα δεξιά).
#define row1 0
#define row2 1
#define row3 2
#define row4 3
#define row5 4
#define row6 5
#define col1 8
#define col2 9
#define col3 10
#define col4 11
#define col5 12
#define col6 13

int row_num = 0; // Η γραμμή του matrix η οποία είναι αναμμένη κάθε φορά (μέθοδος πολυπλεξίας)

boolean matrix[6][6]; // Κάθε στοιχείου του πίνακα matrix αντιπροσωπεύει ένα LED (0-σβηστό, 1-αναμμένο)

// Ορισμός της απεικόνισης των αριθμητικών ψηφίων 0-9
byte dig[][6] = {{28,34,34,34,34,28},
{8,8,8,8,8,8},
{28,34,16,8,4,62},
{28,34,32,24,34,28},
{24,20,18,62,16,16},
{62,2,2,30,32,30},
```

```
{28,2,2,30,34,28},
{62,32,32,16,8,8},
{28,34,34,28,34,28},
{28,34,34,60,32,28}
};

// Ορισμός της απεικόνισης του αναμμένου και του σβηστού LED Matrix
byte setMat[6] = {63,63,63,63,63,63};
byte clearMat[6] = {0,0,0,0,0,0};

void setup() {
  // Ορισμός των θυρών 0-5 και 8-13 σαν εξόδους
  pinMode(row1, OUTPUT);
  pinMode(row2, OUTPUT);
  pinMode(row3, OUTPUT);
  pinMode(row4, OUTPUT);
  pinMode(row5, OUTPUT);
  pinMode(row6, OUTPUT);
  pinMode(col1, OUTPUT);
  pinMode(col2, OUTPUT);
  pinMode(col3, OUTPUT);
  pinMode(col4, OUTPUT);
  pinMode(col5, OUTPUT);
  pinMode(col6, OUTPUT);

  // Καθορισμός των παραμέτρων της διακοπής χρονισμού - Timer Interrupt
  // Χρησιμοποιείται ο Timer1 για τη διακοπή ανά τακτά χρονικά
διαστήματα
  cli();                    // Απενεργοποίηση των διακοπών
  TCCR1A = 0;               // Μηδενισμός όλων των flag στον καταχωρητή
TCCR1A
  TCCR1B = 0;               // Μηδενισμός όλων των flag στον καταχωρητή
TCCR1B
  TCNT1  = 0;               // Αρχική τιμή του Timer1
  OCR1A = 10000;            // Τιμή προς σύγκριση με την τιμή του Timer1
  TCCR1B |= (1 << WGM12);   // Χρήση του Timer1
  TCCR1B |= (1 << CS10);    // Όχι prescaler (16 MHz)
  TIMSK1 |= (1 << OCIE1A); // Λειτουργία σύγκρισης
  sei();                    // Ενεργοποίηση των διακοπών
}


void loop() {
  int i;

  // Αναβόσβησε το Matrix 2 φορές
  for (i=0; i<2; i++) {
    setMatrix(setMat);
    delay(500);
    setMatrix(clearMat);
    delay(500);
  }

  // Απεικόνισε τα ψηφία 0-9
  for (i=0; i<10; i++) {
    setMatrix(dig[i]);
    delay(1000);
  }

  // Σβήσε το Matrix
  setMatrix(clearMat);
```

```
  delay(1000);
}

// Ρουτίνα εξυπηρέτησης της διακοπής
// Ενεργοποιεί τις γραμμές με τη σειρά και καθορίζει ποιες στήλες θα
πρέπει να είναι αναμμένες
ISR(TIMER1_COMPA_vect){
  int i;
  digitalWrite(row_num, LOW);
  row_num++;
  if (row_num > 5) row_num = 0;
  for (i=0; i<6; i++)
    digitalWrite(i+8, matrix[row_num][i]);
  digitalWrite(row_num, HIGH);
}

// Ρουτίνα αλλαγής της απεικόνισης του LED Matrix
void setMatrix(byte mVal[6]) {
  int i,j;
  byte mask;

  for(i=0; i<6; i++) {
    mask = B1;
    for (j=0; j<6; j++) {
      matrix[i][j] = mask & mVal[i];
      mask *= 2;
    }
  }
}
```

Σχόλια - Ανάλυση του κώδικα

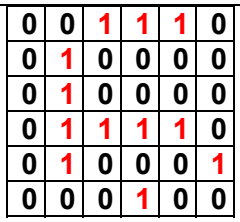
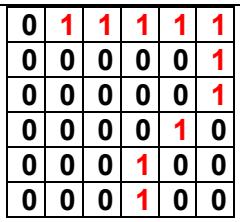
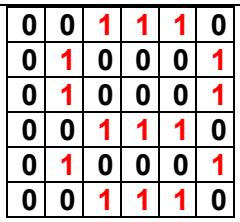
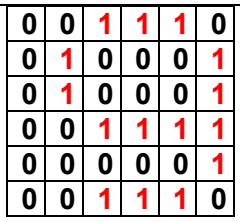
- Οι θύρες 0 έως 5 και 8 έως 13 δηλώνονται ως θύρες εξόδου.
- Ο πίνακας Boolean matrix[6][6] αντιπροσωπεύει κάθε ένα LED με μια Boolean τιμή. Εάν κάποιο στοιχείο αυτού του πίνακα έχει την τιμή 1, τότε το αντίστοιχο LED είναι αναμμένο. Το αντίθετο ισχύει για την τιμή 0.
- Ο πίνακας dig περιγράφει ανά γραμμή το πώς θα απεικονιστούν τα ψηφία 0 έως 9 (βλέπε επόμενη ενότητα).
- Οι πίνακες setMat και clearMat ορίζουν, ανά γραμμή, το πλήρες αναμμένο και σβηστό LED Matrix αντίστοιχα.
- Το πρόγραμμα χρησιμοποιεί τον Timer 1 για να δημιουργεί διακοπές ανά τακτά χρονικά διαστήματα, ώστε να ανάβει η επόμενη γραμμή. Η ρουτίνα που επυπηρετεί τη διακοπή είναι η ISR(TIMER1_COMPA_vect). Η ISR εκτός του να σβήνει την γραμμή που είναι αναμμένη για να ανάψει την επόμενη, καθορίζει και τα LED που θα πρέπει να είναι αναμμένα κάθε φορά σε κάθε γραμμή.
- Η ρουτίνα setMatrix αλλάζει τα δεδομένα του πίνακα matrix ανάλογα με το όρισμα που της δίνουμε (Πίνακας με 6 στοιχεία. Κάθε στοιχείο καθορίζει τις τιμές κάθε γραμμής του matrix πίνακα).

Απεικόνιση στο LED Matrix

Το Matrix μπορεί να απεικονίσει οτιδήποτε με μέγεθος 6Χ6. Στη συγκεκριμένη παρουσίαση είναι προγραμματισμένο να αναβοσβήνει 2 φορές και να απεικονίζει τα αριθμητικά ψηφία 0 έως 9 ανά 1 δευτερόλεπτο.

Για κάθε απεικόνιση χρησιμοποιείται ένας πίνακας με 6 στοιχεία. Κάθε στοιχείο αντιπροσωπεύει και μια γραμμή του Matrix κωδικοποιημένη έτσι ώστε κάθε LED της γραμμής να αντιπροσωπεύει ένα δυαδικό ψηφίο. Επομένως κάθε γραμμή αντιστοιχεί σε έναν αριθμό 6 bit. Μια πλήρως αναμμένη γραμμή θα έχει την τιμή 63 (πίνακας setMat), ενώ μια πλήρως σβηστή θα έχει την τιμή 0 (πίνακας clearMat).
Ο υπολογισμός της απεικόνισης των αριθμητικών ψηφίων 0 έως 9 (πίνακας dig) δίνεται στο παρακάτω σχήμα.

Ψηφίο	1 2 4 8 16 32	Κωδικοποίηση
0	0 0 1 1 1 0 0 1 0 0 0 1 0 1 0 0 0 1 0 1 0 0 0 1 0 1 0 0 0 1 0 0 1 1 1 0	28 34 34 34 34 28
1	0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0	8 8 8 8 8 8
2	0 0 1 1 1 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 1 1 1 1	28 34 16 8 4 62
3	0 1 1 1 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1 0 0 1 0 0 0 1 0 0 1 1 1 0	28 34 32 24 34 28
4	0 0 0 1 1 0 0 0 1 0 1 0 0 1 0 0 1 0 0 1 1 1 1 1 0 0 0 0 1 0 0 0 0 0 1 0	24 20 18 62 16 16
5	0 1 1 1 1 1 0 1 0 0 0 0 0 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 1 0 1 1 1 1 0	62 2 2 30 32 30

6		28 2 2 30 34 8
7		62 32 32 16 8 8
8		28 34 34 28 34 28
9		28 34 34 60 32 28

Εφαρμογή 7: Ανίχνευση κίνησης με PIR sensor

Συναγερμός με τον Arduino

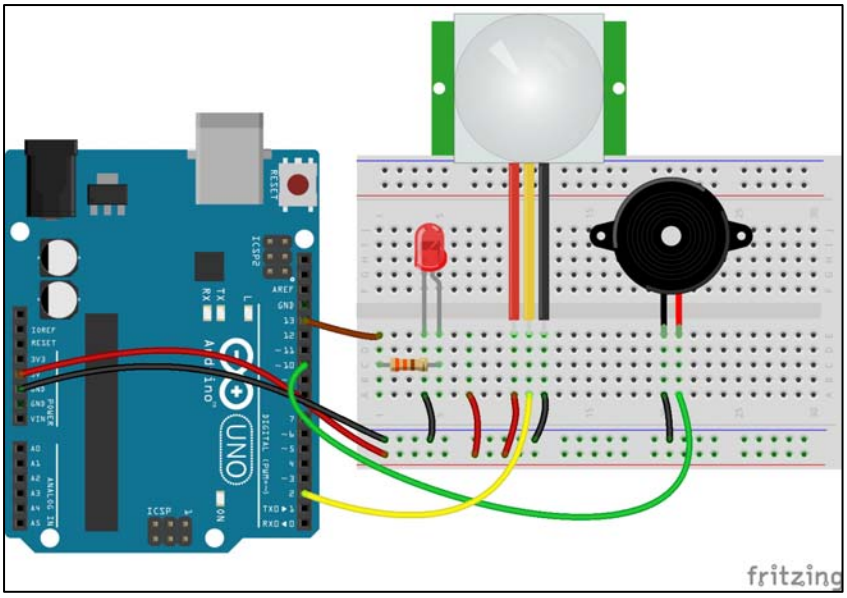
Α. Μαρμαρινός

Υλικά

Τα υλικά που χρειάστηκαν για αυτή την εφαρμογή, είναι:
-1 Arduino Uno
-1 LED και μια αντίσταση 330Ω
-1 αισθητήρας PIR
-1 βομβητής (buzzer) ή ένα μεγάφωνο (speaker)

Συνδεσμολογία με το Arduino

Ο αισθητήρας κίνησης διαθέτει τρεις ακροδέκτες εκ των οποίων οι δύο είναι για την τροφοδοσία (5V, GND) και ο τρίτος είναι η έξοδος. Η έξοδος του αισθητήρα είναι ψηφιακή και γίνεται HIGH όταν ανιχνευθεί κίνηση και LOW όταν δεν ανιχνευθεί κίνηση. Συνδέεται στην ψηφιακή θύρα του arduino (pin2) και απεικονίζεται με το κίτρινο καλώδιο
Το ενδεικτικό LED συνδέεται επίσης στην ψηφιακή θύρα του arduino (pin13), με ή χωρίς αντίσταση περιορισμού ρεύματος (330Ω), ενώ ο βομβητής συνδέεται σε μια άλλη ελεύθερη πόρτα με δυνατότητα PWM (pin10).



Πρόγραμμα

```
1  int pir = 0;                  // μεταβλητή για την ανάγνωση της εισόδου
2  int inputPin = 2;             // επιλογή του pin σύνδεσης του αισθητήρα PIR
3  int ledPin = 13;              // επιλογή του pin σύνδεσης του LED
4  int speakerPin = 10;          // επιλογή του pin σύνδεσης του Buzzer

5  void setup() {
6    pinMode(inputPin, INPUT);    // ορισμός του inputPin ως είσοδο
7    pinMode(ledPin, OUTPUT);     // ορισμός του ledPin ως έξοδο
8    pinMode(speakerPin, OUTPUT);// ορισμός του speakerPin ως έξοδο
9  }

10 void loop(){
11   pir = digitalRead(inputPin);  // ανάγνωση της εισόδου
12   if (pir == HIGH) {            // αν έχει ανιχνευθεί κίνηση
13      digitalWrite(ledPin, HIGH);  // άναψε το LED
14      tone(speakerPin, 300, 300);  // χτύπα συναγερμό
15   }
16   else {                        // αλλιώς
17      digitalWrite(ledPin, LOW);   // σβήσε το LED
18   }
19   delay(500);                   // καθυστέρηση 500ms
20 }
```

Σχόλια - Ανάλυση του κώδικα

Πολύ απλή εφαρμογή επίδειξης στην οποία η έξοδος ενός αισθητήρα κίνησης διαβάζεται από τον arduino και αν είναι σε κατάσταση HIGH, δηλαδή έχει ανιχνευθεί κίνηση, ανάβει ένα LED ενώ ταυτόχρονα ηχεί ο βομβητής. Το πρόγραμμα είναι απλό, μπορεί να είναι μια από τις αρχικές ασκήσεις σας με τους μαθητές σας μετά την κλασική άσκηση που αναβοσβήνει ένα LED (blink).

Η λειτουργία φαίνεται και από τα σχόλια που συνοδεύουν τον κώδικα.

- Στις εντολές με αρίθμηση 1 ως 4 δηλώνονται οι μεταβλητές του προγράμματος
- Στις εντολές με αρίθμηση 5 ως 10 δηλώνεται ο κάθε ακροδέκτης ως είσοδος ή έξοδος
- Στις εντολές με αρίθμηση 10 ως 20 είναι το κυρίως πρόγραμμα, το οποίο επαναλαμβάνεται συνέχεια. Στο τμήμα αυτό, διαβάζεται η κατάσταση του αισθητήρα και γίνονται οι αντίστοιχες ενέργειες.

Άλλες πληροφορίες - Εφαρμογές

Ο αισθητήρας PIR έχει εμβέλεια μερικά μέτρα (ο συγκεκριμένος τύπος, 8m) και μπορεί να τοποθετηθεί σε κάποιο χώρο ώστε να ανιχνεύει κίνηση. Επειδή ο αισθητήρας αυτός ανιχνεύει υπέρυθρη ακτινοβολία που εκπέμπεται από θερμά σώματα, δεν θα ανιχνεύσει κίνηση αντικειμένων που βρίσκονται σε θερμοκρασία ίδια με τη θερμοκρασία περιβάλλοντος. Ακόμη περιέχει μέσα του δύο αισθητήρες (και αρκετά κυκλώματα) ώστε να συγκρίνει την ακτινοβολία που δέχεται σε αυτούς και μόνο όταν είναι διαφορετική να δίνει έξοδο HIGH. Αυτό σημαίνει ότι δεν ανιχνεύει την ύπαρξη ενός σώματος στο χώρο αλλά μόνο την μετακίνησή του.

Ο συγκεκριμένος τύπος διαθέτει δύο ποτενσιόμετρα ρύθμισης: ένα για τη ρύθμιση της εμβέλειας και ένα για τη ρύθμιση του χρόνου που θα παραμένει σε κατάσταση HIGH μετά τη διέγερσή του.

Τυπική εφαρμογή είναι σε οικιακά αλλά και επαγγελματικά συστήματα συναγερμού.

Άλλες πιθανές εφαρμογές του κυκλώματος:

- Ειδοποίηση εισόδου πελάτη σε κατάστημα
- Αυτόματο άνοιγμα φωτιστικών όταν κάποιος εισέρχεται σε έναν χώρο (σε συνδυασμό με αισθητήρα φωτός)
- Τοποθέτηση στο ντουλάπι με τα γλυκά ώστε να γίνεται αντιληπτή κάθε προσπάθεια παραβίασης του χώρου από τους συνήθεις υπόπτους

και ότι άλλο μπορεί να φανταστεί κάποιος.

8η Εφαρμογή: Αυτοματισμός με LDR

Αυτοματισμός για έλεγχο ανάμματος LED (ή οποιουδήποτε στοιχείου φωτισμού) σε σχέση με τον περιβάλλοντα φωτισμό, με μεταβλητό-ρυθμιζόμενο σημείο έναυσης (threshold) και ταυτόχρονη απεικόνιση των πληροφοριών σε LCD Display.

Χ. Τουμασάτος

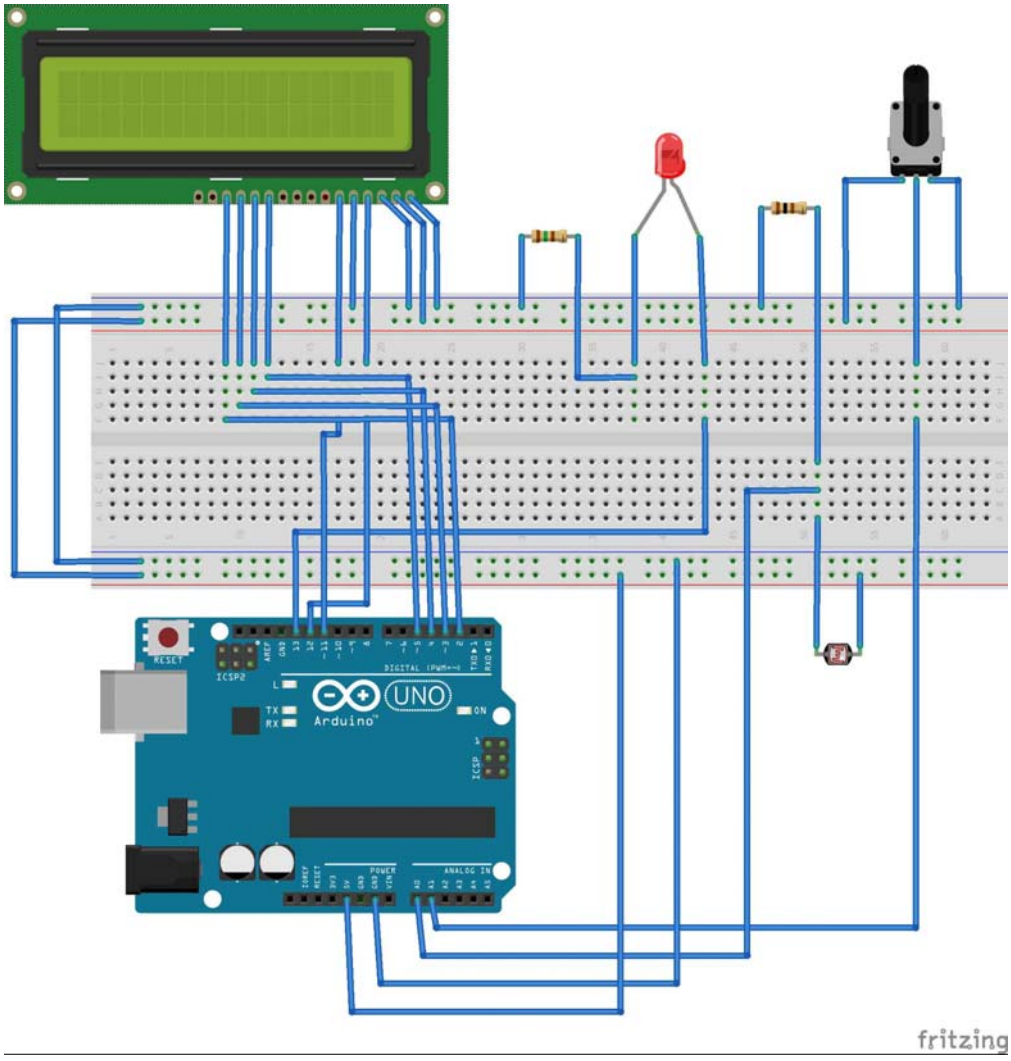
Υλικά

Τα υλικά που χρειάστηκαν για αυτή την εφαρμογή, είναι:

-Ένα Arduino Uno
-Ένα Led
-USB καλώδιο
-2 αντιστάσεις (για το LED και το LDR)
-Display (για απεικόνιση πληροφοριών)
-Ποτενσιόμετρο (για καθορισμό του threshold)
-LDR (sensor φωτός)

Συνδεσμολογία με το Arduino

Για να συνδεθεί σωστά το κύκλωμα πρέπει να συνδεθεί το μακρύ ποδαράκι του Led στο pin13 και το κοντό ποδαράκι στη γείωση (GND). Συνήθως, οι περισσότερες πλακέτες έχουν ήδη ένα Led ενσωματωμένο.



Πρόγραμμα

Arduino Code:

```
#include <LiquidCrystal.h>  //δήλωση ότι συνδέουμε LCD
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);//δήλωση εξόδων προς το lcd

int sensorPin = A0;   //δήλωση σημείου ένωσης LDR
int thresholdPin = A1;  //δήλωση σημείου ένωσης ποτενσιόμετρου
int sensorValue; //δήλωση τιμής εξόδου LDR (μεταβλητή)
int thresholdValue; //δήλωση μεταβλητής threshold (σημείο έναυσης)
int ledPin = 13;              //δήλωση σημείου σύνδεσης του LED

void setup() {
  pinMode(ledPin, OUTPUT);      //ρυθμίζει το pin 13 ως έξοδο
  lcd.begin(16, 2); //δήλωση πλήθους ψηφίων και γραμμών του lcd
}




void loop() {
  thresholdValue = analogRead(thresholdPin); //ανάγνωση ποτενσιόμετρου
  sensorValue = analogRead(sensorPin); //ανάγνωση LDR
  lcd.setCursor(0, 0); //ορισμός αρχικού γράμματος στο display
  lcd.print("T:"); //εμφάνιση στο display του «T:» (για threshold)
  lcd.print(sensorValue); //εμφάνιση στο display του «sensorvalue»
  lcd.print("   "); //εμφάνιση μερικών space στο display
  lcd.setCursor(8, 0);  //μεταφορά κέρσορα οθόνης στη θέση 8,1η γραμμή
  lcd.print("O:"); //εμφάνιση στο display του «O:»
  lcd.print(thresholdValue); //εμφάνιση στο display της τιμής threshold
  lcd.print("   "); //εμφάνιση στο display λίγων ακόμα κενών


  lcd.setCursor(0, 1); //μεταφορά κέρσορα οθόνης στη θέση 0,2η γραμμή
  if (sensorValue <= thresholdValue) { //αν το LDR δίνει<=thr ανάβει το led
    digitalWrite(ledPin, HIGH); //βάζει το LED σε κατάσταση on
    lcd.print("LED Anoikto");  //εμφάνιση στο display"Led anoikto"
  }
  else { //αν το LDR δίνει τιμή > threshold, το led σβηνει
    digitalWrite(ledPin, LOW);  //βάζει το LED σε κατάσταση off

    lcd.print("LED Kleisto");  //εμφάνιση στο display"Led kleisto"
  }

  delay(500); //αναμονή για 0,5 sec και ξεκινά το πρόγραμμα από την αρχη
}
```

Σχόλια-Ανάλυση του κώδικα

Το νόημα της κατασκευής είναι να ανάβει αυτόματα ένα LED (ή μια συστοιχία φωτισμού αντικαθιστώντας το LED με ρελέ) με το που διαγνώσει το LDR πτώση στο φωτισμό του περιβάλλοντος χώρου. Επιπρόσθετα, με το ποτενσιόμετρο μπορούμε να καθορίσουμε εμείς το σημείο της εντολής έναυσης (threshold) δηλαδή σε «πόσο περιβάλλοντα φωτισμό» θα δοθεί η εντολή. Οι πληροφορίες, (σημείο threshold, LED ON, LED OFF) θα εμφανίζονται σε display.

Αρχικά δηλώνουμε την ύπαρξη του LCD και σε ποιες εξόδους συνδέεσαι. Αφού δηλώσουμε το πού συνδέεται ο αισθητήρας LDR και το ποτενσιόμετρο, πού ρυθμίζουμε το σημείο έναυσης του LED και πού συνδέεται το LED, δηλώνουμε και τις δυο μεταβλητές για τις τιμές του LDR και threshold και ορίζουμε τη διάσταση του LCD.
Μετά ξεκινά το επαναλαμβανόμενο κομμάτι του κώδικα κατά το οποίο διαβάζουμε τις τιμές του ποτενσιόμετρου και του LDR και εμφανίζουμε στην οθόνη τα γράμματα «T:» για το LDR και την τιμή του και «O:» για το threshold και την τιμή του.
Μετά ξεκινά η σύγκριση των δυο τιμών και εάν η τιμή του LDR είναι μικρότερη ή ίση του threshold, το led ανάβει και εμφανίζει στην οθόνη «led anoikto» εάν όχι το led σβήνει και στην οθόνη εμφανίζει «led kleisto». Μετά από μια καθυστέρηση μισού δευτερολέπτου επαναλαμβάνεται ο παραπάνω έλεγχος.

Ενδιαφέρουσα ενέργεια θα ήταν εάν πλησιάσουμε το LDR στο LED όπου παρατηρούμε το LED να αναβοσβήνει γιατί το LED φωτίζει το LDR και «αναιρεί τον εαυτό του» αφού μόλις ανάψει το LED, φωτίζει το ldr και στον επόμενο κύκλο προγράμματος η εντολή είναι να σβήσει το LED!

9η Εφαρμογή: Αναπαραγωγή μελωδίας

Δημιουργία μουσικών τόνων με τον Arduino

Ι. Μπάκας

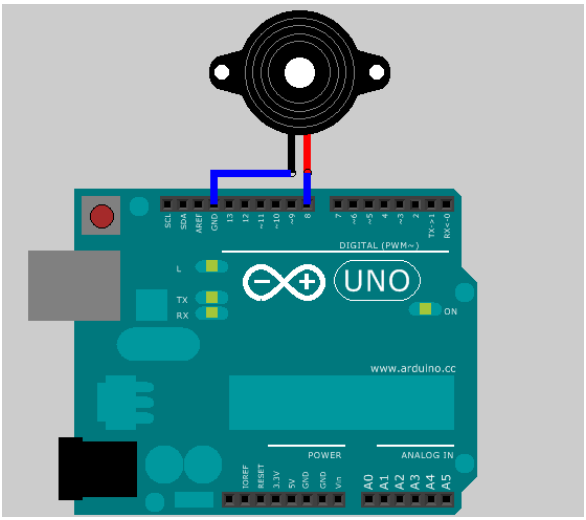
Υλικά :

Τα υλικά που θα χρησιμοποιηθούν στην εφαρμογή μας είναι:

- Ένα Arduino Uno
- Ένα μεγάφωνο 8Ωμ
- Καλώδιο USB

Κύκλωμα

Η συνδεσμολογία του κυκλώματος φαίνεται στο παρακάτω σχήμα.



Τον θετικό πόλο του μεγάφωνου τον συνδέουμε στο pin 8 του Arduino ενώ τον αρνητικό πόλο του μεγαφώνου τον συνδέουμε στη γείωση (GND). Το μεγάφωνο για να έχει καλύτερη απόδοση μπορούμε να το τοποθετήσουμε μέσα σε ένα κουτί που θα παίζει το ρόλο του ηχείου.

Πρόγραμμα

```
/*
 Το πρόγραμμα αναπαράγει μια μελωδία μια φορά.
 http://arduino.cc/en/Tutorial/Tone
*/
 #include "pitches.h" // Συμπεριέλαβε στον κώδικα την βιβλιοθήκη
pitches.h
// Οι νότες της μελωδίας, το 0 αντιστοιχεί σε παύση:
int melody[] = {NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3,0, NOTE_B3,
NOTE_C4};

// Η διάρκεια της κάθε νότας : 4 = τέταρτο, 8 = όγδοο , κλπ.:
int noteDurations[] = {4,8,8,4,4,4,4,4};

void setup()
{
// επανέλαβε τον παρακάτω κώδικα για κάθε νότα της μελωδίας:
for (int thisNote = 0; thisNote < 8; thisNote++) //αρχή επανάληψης.
      {

// Για να υπολογίσουμε τη διάρκεια της νότας διαιρούμε το 1
// δευτερόλεπτο (1000 msec) με τον αριθμό που δηλώνει τη διάρκεια
// της νότας. Πχ τέταρτο της νότας = 1000/4 κλπ.
      int noteDuration = 1000/noteDurations[thisNote];
```

```
// παίξε τη νότα όσο είναι η διάρκεια της.
      tone(8, melody[thisNote],noteDuration);

//για να ξεχωρίζουν οι νότες, βάζουμε ένα ηχητικό κενό ανάμεσά τους. Το
κενό αυτό είναι το 30% //της διάρκειας της νότας. Άρα ο συνολικός
χρόνος αναμονής (παίξιμο της νότας + ηχητικό κενό) //είναι η διάρκεια
της νότας X 1.3.
      int pauseBetweenNotes = noteDuration * 1.30;//Υπολόγισε την
                                                  //καθυστέρηση
      delay(pauseBetweenNotes);    //Καθυστέρηση

// Σταμάτα να παίζεις τη νότα.
      noTone(8);

      }     //τέλος επανάληψης
}
void loop() {
  // δεν υπάρχει κώδικας που χρειάζεται να επαναληφθεί.
}
```

Ανάλυση του κώδικα:

Αρχικά δηλώνουμε 2 μεταβλητές σε μορφή πίνακα.

Η μεταβλητή `int melody[]` περιέχει τις νότες της μελωδίας με τη σειρά που θα παιχτούν. (πχ NOTE_C4 = ΝΤΟ, NOTE_G3 = ΣΟΛ κλπ)

Η μεταβλητή `int noteDurations[]` περιέχει τις διάρκειες των νοτών της προηγούμενης μεταβλητής (πχ 4=τέταρτο, 8=όγδοο κλπ)

Στο βασικό κορμό του προγράμματος `void setup()` γράφουμε το πρόγραμμα που θα αναπαράγει τη μελωδία.

Αρχικά δημιουργούμε μια δομή επανάληψης η οποία επαναλαμβάνει τον κώδικα που περιέχει 8 φορές (όσες είναι και οι νότες της μελωδίας):

`for (int thisNote = 0; thisNote < 8; thisNote++)`

Η μεταβλητή `thisNote` παίρνει τιμές από 0 έως 7.

Στη συνέχεια υπολογίζουμε τον πραγματικό χρόνο αναπαραγωγής μιας νότας χρησιμοποιώντας την μεταβλητή `noteDuration`. Ο πραγματικός χρόνος υπολογίζεται από τον τύπο 1000/t , όπου t ο αριθμός που δηλώσαμε στη μεταβλητή `noteDurations[]` .

Δίνουμε εντολή να ξεκινήσει η αναπαραγωγή της νότας χρησιμοποιώντας τη συνάρτηση: `tone()`

Η συνάρτηση αυτή δημιουργεί σε έναν ακροδέκτη του Arduino μια τετραγωνική κυματοφορφή με DC=50%. Η συνάρτηση έχει την παρακάτω σύνταξη:

`Tone(pin, frequency, duration)`

Όπου pin= ο αριθμός του ακροδέκτη, frequency= συχνότητα σε Hz , duration=διάρκεια αναπαραγωγής σε msec.

Στη συνέχεια υπολογίζουμε την μεταβλητή `int pauseBetweenNotes` η οποία δηλώνει το χρόνο σε msec που διαρκεί μια νότα συν ένα ηχητικό χρονικό κενό πριν παιχτεί η επόμενη νότα. Το ηχητικό κενό είναι το 30% της διάρκειας της νότας.

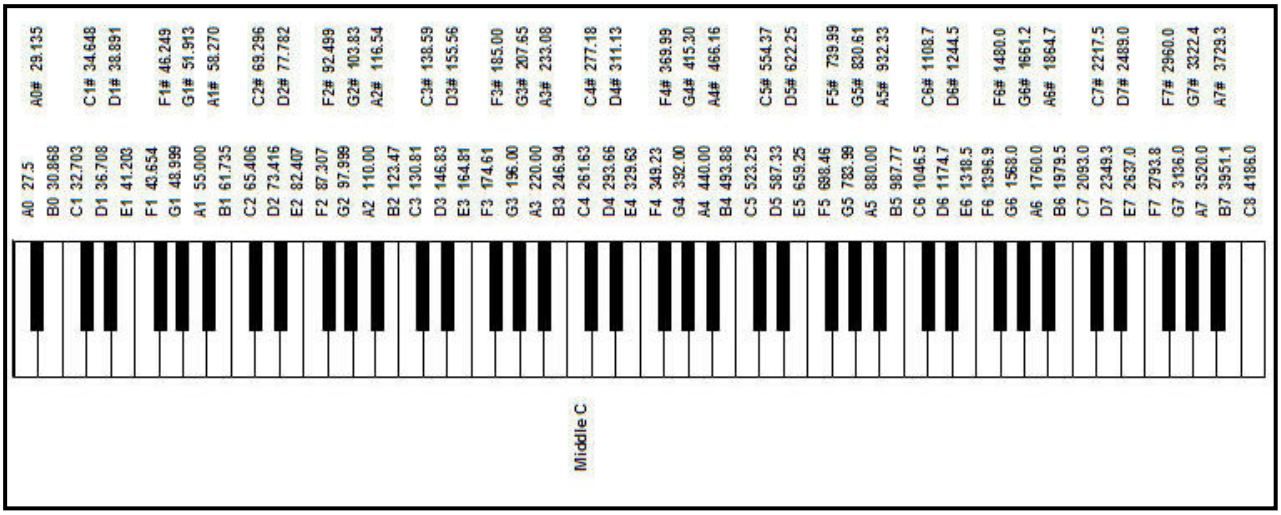
Η συνάρτηση `delay()` δημιουργεί μια χρονική καθυστέρηση όσο είναι η προηγούμενη μεταβλητή.

Στη συνέχεια η συνάρτηση `noTone(pin)` σταματάει τη δημιουργία της κυματομορφής στο αντίστοιχο pin.

Τέλος ο βρόγχος επανάληψης ξεκινάει από την αρχή και αναπαράγεται η επόμενη νότα.

Υποσημείωση:

Στην παρακάτω εικόνα φαίνεται η αντιστοιχία που έχουν οι νότες του πενταγράμμου με τις συχνότητες τις οποίες αναπαράγουν.



- Αντιστοιχία συμβόλου – νότας:

  C=ΝΤΟ, D= ΡΕ, E=ΜΙ, F=ΦΑ, G=ΣΟΛ, A=ΛΑ, B=ΣΙ

- Ο αριθμός δίπλα στο σύμβολο δηλώνει τον αριθμό της οκτάβας πχ C3 = η ΝΤΟ της 3^{ης} οκτάβας.
- Το σύμβολο # είναι η δίεση πχ F#4. Στο πρόγραμμα η νότα αυτή δηλώνεται ως GS3_NOTE
- Η βιβλιοθήκη pitches.h περιέχει τις αντιστοιχίες νότας - συχνότητας, πχ G3_NOTE = 196 Hz