1ο ΕΠΑΛ- Ε.Κ. Συκεών -Τομέας: Ηλεκτρονικής, Ηλεκτρολογίας και Αυτοματισμού
Εκπαιδευτικοί: Μπουλταδάκης Στέλιος – Μαυρίδης Κωνσταντίνος

Μάθημα: Αισθητήρες-Ενεργοποιητές: Έλεγχος **STEPPER MOTOR**

Στόχοι:
α) κατανόηση του προγραμματισμού με κλάσεις-βιβλιοθήκες όπως η Servo και με αντικείμενα της κλάσης
β) κατανόηση του τρόπου λειτουργίας των βηματικών κινητήρων
Βιβλιογραφία:

1) Απόστολος Σεργιάδης, Διπλωματική Εργασία, Τμήμα Ηλεκρολόγων Μηχανικών καιΜηχανικών Η/Υ, ΕΜΠ, 2009
2) http://www.tigoe.com/pcomp/code/circuits/motors/stepper-motors/
3) https://www.instructables.com/id/BYJ48-Stepper-Motor/
   By Mohannad Rawashdeh in TechnologyArduino

Τα Υλικά που θα χρειαστούμε:
1) Arduino board .
2) Ένα βηματικό κινητήρα BYJ48 Stepper Motor που λειτουργεί με 5 Volts
3) Την πλακέτα οδηγού ULN2003 Stepper Motor driver Module
4) Ενα βραχυκυκλωτήρα Jumper
5) Τροφοδοσία 5Volts (voltage source "Optional") . Είναι προτιμότερη η εξωτερική τροφοδοσία του μοτέρ με σταθεροποιημένο τροφοδοτικό 5 Volts αντί της παροχής τροφοδοσίας του Arduino UNO R3

Το Αντικείμενο της άσκησης:
Α) εξοικείωση των μαθητών με προχωρημένες εντολές σειριακής επικοινωνίας δεδομένων μεταξύ του Arduino και των εφαρμογών σε Η/Υ και πιο συγκεκριμένα με την εφαρμογή: Serial Monitor από το περιβάλλον προγραμματισμού του Arduino και με εφαρμογή που θα αναπτύξουν οι μαθητές με το λογισμικό LabView
Β) εξοικείωση των μαθητών με τη λειτουργία ενός βηματικού κινητήρα και με τον έλεγχο της περιστροφής του κινητήρα από το Arduino

Βήμα 1^{ο}: Τι πρέπει να γνωρίζουμε:

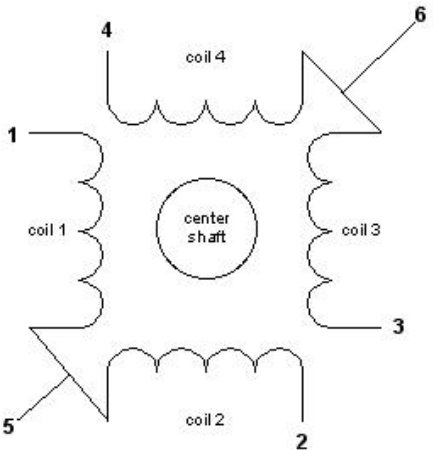
Οι βηματικοί κινητήρες διαφέρουν από τους άλλους τύπους κινητήρων συνεχούς και εναλλασσομένου ρεύματος στο ότι τροφοδοτούνται με παλμούς και παράγουν ηλεκτρική κίνηση. Ο άξονας τους δεν έχει μια συνεχή περιστροφική κίνηση, αλλά περιστρέφεται κατά μία γωνία κάθε φορά που δέχεται ένα παλμό. Ένας βηματικός κινητήρας είναι ένας κινητήρας ελεγχόμενος από μια σειρά ηλεκτρομαγνητικών σπειρών. Ο κεντρικός άξονας έχει μια σειρά από μαγνήτες προσαρμοσμένους πάνω του και πηνία που τον περιβάλλουν. Στα πηνία δίδεται διαδοχικά ηλεκτρικό ρεύμα ή όχι, δημιουργώντας μαγνητικά πεδία τα οποία απωθούν ή έλκουν τους μαγνήτες του άξονα, προκαλώντας την περιστροφή του κινητήρα.

Αυτός ο σχεδιασμός επιτρέπει τον πολύ ακριβή έλεγχο του κινητήρα: με κατάλληλη παλμική κίνηση, μπορεί να γυρίσει σε πολύ ακριβή βήματα. Χρησιμοποιούνται σε εκτυπωτές, δίσκους και άλλες συσκευές όπου απαιτείται ακριβής μετακίνηση του κινητήρα.

Υπάρχουν δύο βασικοί τύποι βηματικών κινητήρων, οι μονοπολικοί βηματικοί κινητήρες και οι διπολικοί βηματικοί κινητήρες.

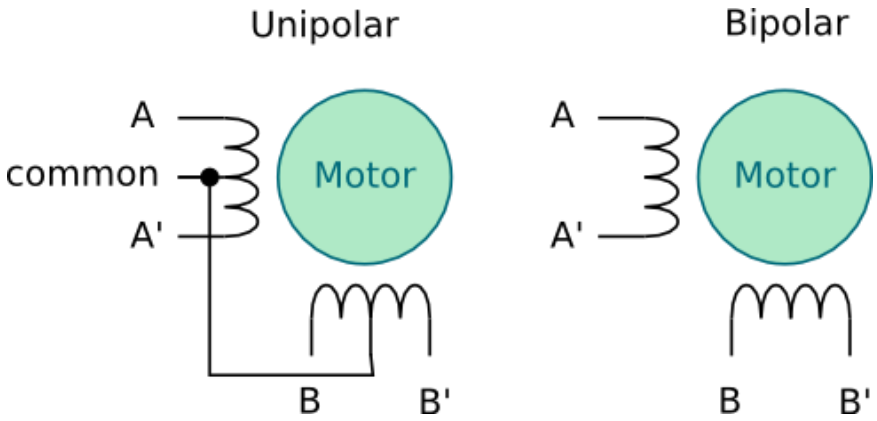
Μονοπολικοί βηματικοί κινητήρες

Ο μονοπολικός βηματικός κινητήρας έχει πέντε ή έξι σύρματα και τέσσερα πηνία (στην πραγματικότητα δύο πηνία διαιρούμενα με κεντρικές συνδέσεις σε κάθε πηνίο). Οι κεντρικές συνδέσεις των πηνίων συνδέονται μεταξύ τους και χρησιμοποιούνται ως σύνδεση ρεύματος. Ονομάζονται μονοπολικά βηματικά, επειδή η δύναμη πάντα έρχεται σε αυτόν τον ένα πόλο.



Διπολικοί βηματικοί κινητήρες

Ο διπολικός βηματικός κινητήρας έχει συνήθως τέσσερα σύρματα που εξέρχονται από αυτό. Σε αντίθεση με τους μονοπολικούς βηματικούς σταθμούς, οι διπολικοί κινητήρες δεν έχουν κοινή κεντρική σύνδεση. Έχουν δύο ανεξάρτητα σύνολα πηνίων.



Από την ιστοσελίδα:
http://89.22.98.13/pylog/pylog.py?disp=cnt/projects/shapeoko/Turn+a+unipolar+into+a+bipolar+steper.txt

Όπως και οι άλλοι κινητήρες, οι βηματικοί κινητήρες απαιτούν περισσότερη ενέργεια από ό, τι μπορεί να τους δώσει ένας μικροελεγκτής, οπότε θα χρειαστείτε ξεχωριστό τροφοδοτικό για αυτό. Στην ιδανική περίπτωση θα γνωρίζετε την τάση από τον κατασκευαστή. Για την οδήγηση-έλεγχο των βηματικών κινητήρων υπάρχουν πλακέτες-οδηγοί (drivers) ανάλογα με τον κινητήρα και τα χαρακτηριστικά λειτουργίας του. Στο διαδίκτυο υπάρχει μεγάλη ποικιλία οδηγών και βηματικών κινητήρων που συνεργάζονται με τις πλακέτες Arduino UNO όπως ενδεικτικά φαίνεται παρακάτω:

Οδήγηση μονοπολικού βηματικού κινητήρα:

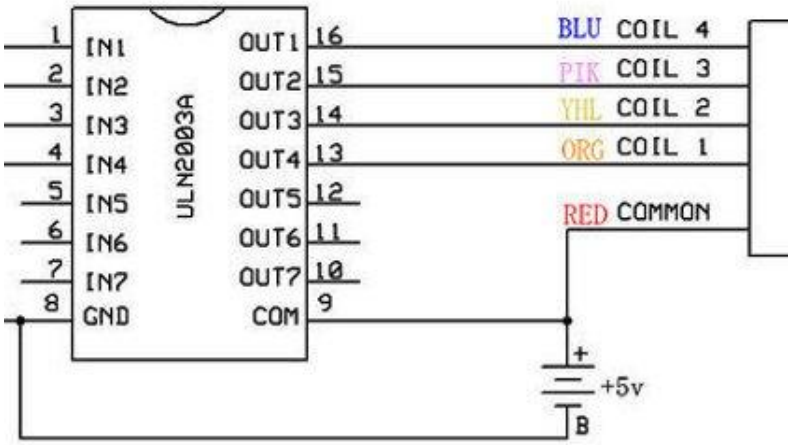
Τύπος κινητήρα: Unipolar Stepper Motor 28-BYJ48 που συνοδεύεται από την πλακέτα οδηγού

Μονάδα οδηγού: ULN2003 Stepper Motor Driver (αν χρησιμοποιήσουμε τον L293 driver θα αφήσουμε το κόκκινο καλώδιο ασύνδετο)

ΤΕΧΝΙΚΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ:

Rated voltage 5VDC, Number of Phase 4, **Speed Variation Ratio 1/64**, **Stride Angle 5.625° /64 Frequency 100Hz**, DC resistance 50Ω±7%, Idle In-traction Frequency > 600Hz,

Idle Out-traction Frequency > 1000Hz , In-traction Torque >34.3mN.m(120Hz), Self-positioning Torque >34.3mN.m, Friction torque 600-1200 gf.cm, Pull in torque 300 gf.cm



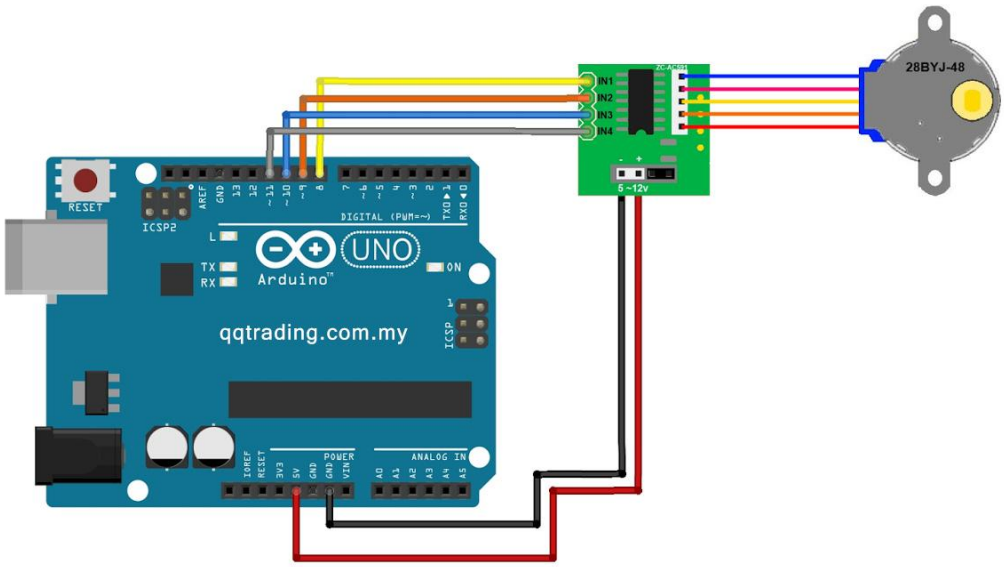
Υπολογισμός βημάτων για την περιστροφή

Το μοτέρ έχει ένα λόγο γραναζιών : Gear ratio=64

ενώ η γωνία που κάνει σε κάθε βήμα (Stride Angle)= 5.625°

Επομένως για να κάνει το **μοτέρ 28-BYJ48** μια πλήρη περιστροφή χρειαζόμαστε **steps**:

steps in One Revolution = 360° / 5.625°= 64
steps = Number of steps in One Revolution * Gear ratio = 64 * 64 =4096 steps



Οι παραπάνω υπολογισμοί είναι διαφορετικοί αν έχουμε έναν άλλο βηματικό κινητήρα όπως για παράδειγμα το **adafruit Stepper Motor** :

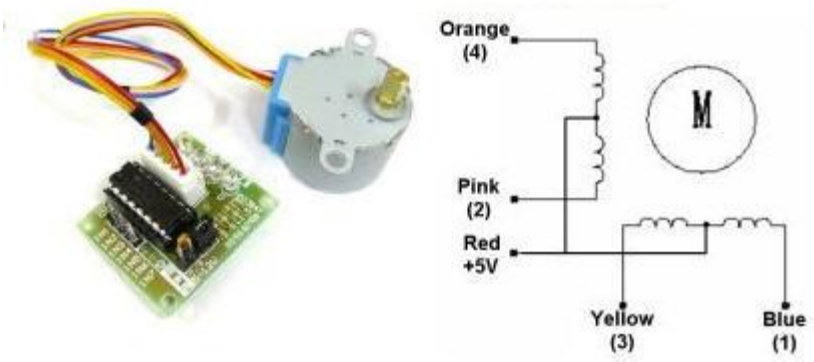
Το μοτέρ αυτό έχει ένα λόγο γραναζιών : Gear ratio=16

ενώ η γωνία που κάνει σε κάθε βήμα (Stride Angle)= 7.5°

Επομένως για να κάνει το **adafruit Stepper Motor** μια πλήρη περιστροφή χρειαζόμαστε **steps**:

steps in One Revolution = 360° / 7.5°= 48
steps = Number of steps in One Revolution * Gear ratio = 48 * 16 =768 steps

Στο τεχνικό εγχειρίδιο για το μοτέρ **28-BYJ48** προτείνεται η οδήγηση μισού-βήματος για την οποία η διαδοχή των εντολών κίνησης φαίνεται στο παρακάτω σχήμα:



Half-Step Switching Sequence

Lead Wire Color	---> CW Direction (1-2 Phase)							
	1	2	3	4	5	6	7	8
4 Orange	-	-						-
3 Yellow		-	-	-				
2 Pink				-	-	-		
1 Blue						-	-	-

Στον κώδικα του Arduino τα παραπάνω βήματα φαίνονται στις εντολές της συνάρτησης Stepper.

Το τμήμα κώδικα για την περιστροφή του μοτέρ βασίζεται στον προτεινόμενο κώδικα από τον ιστότοπο:

https://www.instructables.com/member/Mohannad+Rawashdeh/

που είναι γραμμένος από τον Mohannad Rawashdeh, 28/9/2013

Επιπλέον έχουμε τροποποιήσει τον αριθμό των βημάτων που θα κάνει ο κινητήρας σύμφωνα με την εντολή:

steps_left=rotations*steps_left_per_rev*Flag;

steps_left_per_rev = είναι ο αριθμός βημάτων ανά περιστροφή =4095 (δές παραπάνω)

rotations= ο αριθμός των στροφών που έρχονται από τις εφαρμογές του Η/Υ μέσω σειριακής διασύνδεσης στον Arduino

flag= είναι μια σημαία που ανάλογα με την τιμή της (1 ή 0) κάνει τον κινητήρα να περιστραφεί ή όχι

Επίσης έχουμε προσθέσει την αποστολή του μηνύματος "**End of Rotation**" από τον Arduino στις εφαρμογές του Η/Υ όταν ολοκληρώνεται η περιστροφή του κινητήρα.

Στη συνέχεια δίνονται για ανάλυση με τους μαθητές αποσπάσματα κώδικα που αφορούν τον έλεγχο της σειριακής επικοινωνίας μεταξύ του επικοινωνίας δεδομένων μεταξύ του Arduino και των εφαρμογών σε Η/Υ:

Εκκίνηση και ρυθμίσεις σειριακής επικοινωνίας

```
// Ξεκινά τη σειριακή επικοινωνία:
  Serial.begin(9600);
  while (!Serial) {
    ; // και περιμένει να ανοίξει η θύρα:
  }
```

Αποστολή εισαγωγικού μηνύματος από τον Arduino στις εφαρμογές του Η/Υ (*η επέκταση **In** στην εντολή SerialPrint ωθεί την εφαρμογή Serial Monitor να αλλάξει γραμμή*):

```
 Serial.println();
Serial.println("Give your orders for the stepper motor():");
 Serial.println("F for forward/clockwise, B for backward/anti-clockwise");
```

Ανάγνωση ενός χαρακτήρα σε κάθε loop από το συνολικό μήνυμα που έρχεται στη θύρα USB του Arduino

```
if (Serial.available() > 0)                    // αν φτάσει κάτι στη σειριακή θύρα
 {                                  // το διαβάζει χαρακτήρα προς χαρακτήρα και το κάνει συμβολοσειρά
  while(1)   // επαναλαμβάνει συνεχώς..
  {
   incomingByte = Serial.read();      //διαβάζει ένα χαρακτήρα μετά τον άλλο μέχρι να συναντήσει
                                      //τους χαρακτήρες '\n\'=ASCII(10)=line feed,ASCII(13)=Carriage Return
                                      // που έρχονται με το τέλος κάθε μηνύματος από τον Η/Υ
   if (incomingByte == '\n') break;   // βγαίνει από το while(1) όταν συναντήσει '\n\'
   if (incomingByte == -1) continue;       //  συνεχίζει στο while(1) αν δεν εισαχθούν χαρακτήρες
                                           // (η σειριακή θύρα επιστρέφει -1)
   incomingSerial=String(incomingSerial+incomingByte); // μετατρέπει τους χαρακτήρες που φτάνουν
                                                       // σε συμβολοσειρά (string)
        }
}
```

Εναλλακτικά, ανάγνωση ολόκληρου του μηνύματος (String=πολλοίχαρακτήρες) μήνυμα που έρχεται στη θύρα USB του Arduino

```
if (Serial.available() > 0) {                  // αν φτάσει κάτι στη σειριακή θύρα
   incomingSerial= Serial.readString();   // διάβασε ολόκληρο το μήνυμα ( incoming data) as string
}
```

Έλεγχος του πρώτου χαρακτήρα του μηνύματος που ήρθε από τις εφαρμογές του Η/Υ

```
if (!(incomingSerial.startsWith("F") || incomingSerial.startsWith("B")|| incomingSerial.startsWith("S") ))
     {
      Serial.println("WRONG message. TRY AGAIN ....");
     }
```

Λήψη συγκεκριμένου τμήματος από το μήνυμα που ήρθε από τη σειριακή θύρα:

```
rotationsString=incomingSerial.substring(1);  // παίρνει όλους τους χαρακτήρες μετά τον πρώτο
```

Μετατροπή ενός αλφαριθμητικού δεδομένου σε ακέραιο αριθμό:

```
rotations=rotationsString.toInt();
```

Μετατροπή ενός αλφαριθμητικού δεδομένου σε ακέραιο αριθμό με βάση τις ιδιότητες του πίνακα ASCII:

```
      rotations=0;
       len = rotationsString.length();
       for(i=0; i<len; i++)
        {
          rotations = rotations * 10 + ( rotationsString[i] - '0' );
        }
```

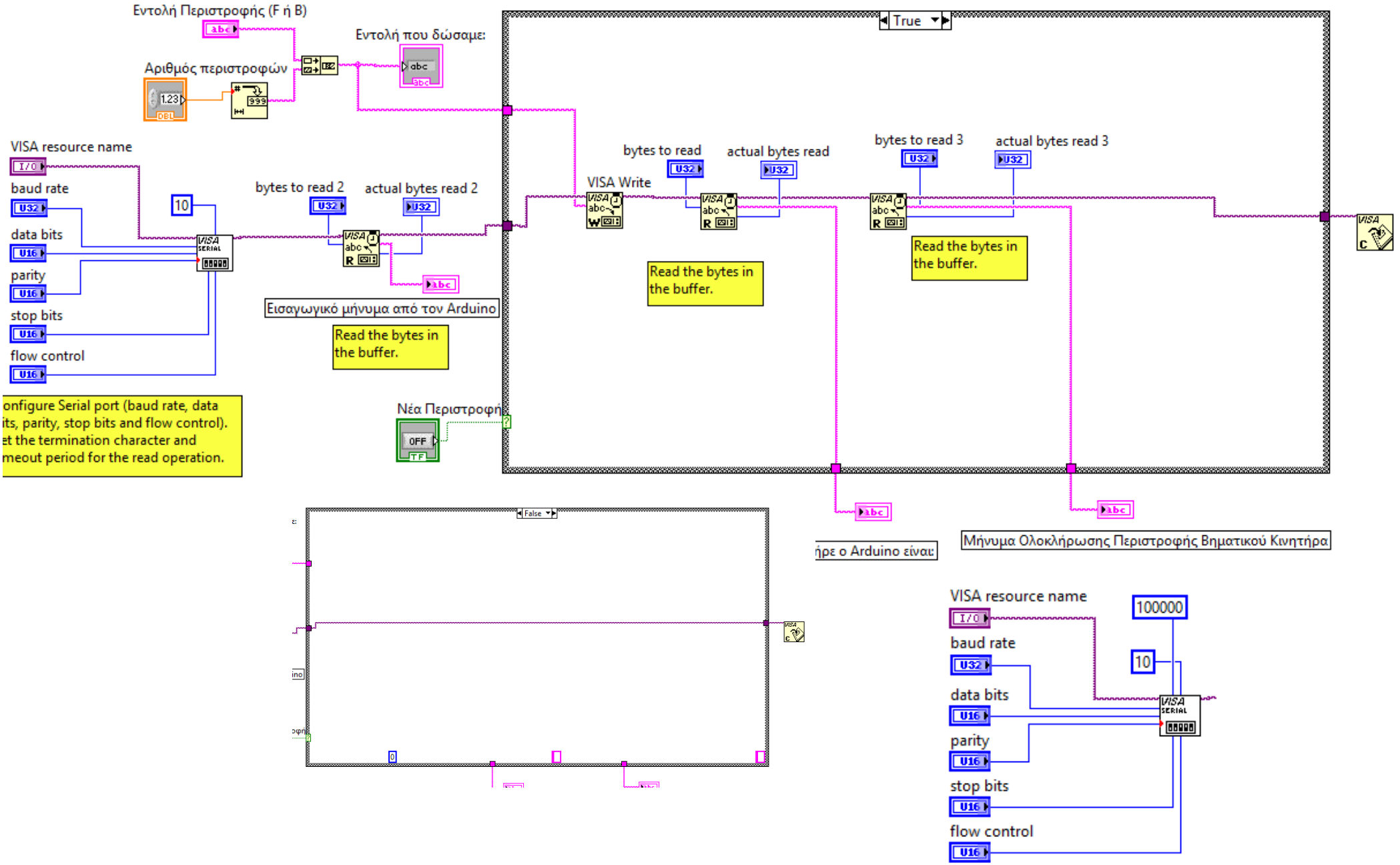
Ο Συνολικός κώδικας που πρέπει να φορτωθεί στον Arduino είναι ο παρακάτω:

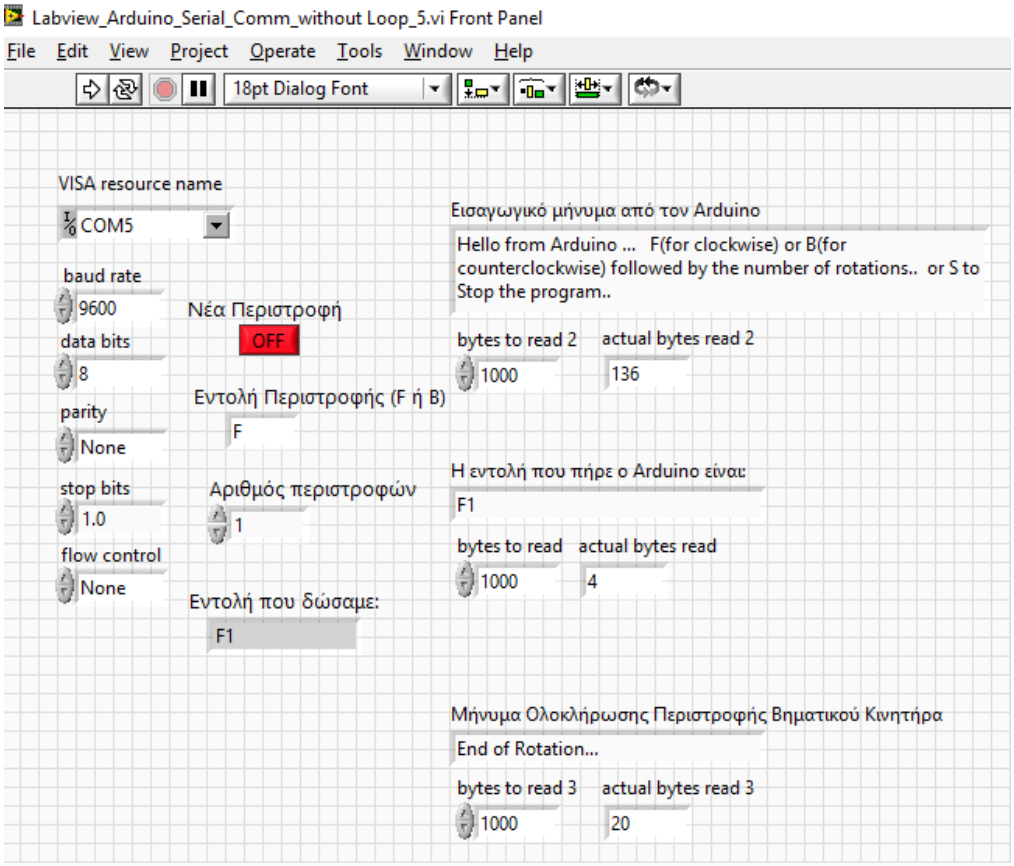
```
// BYJ48 Stepper motor code
  Connect :
  IN1 >> D8
  IN2 >> D9
  IN3 >> D10
  IN4 >> D11
  VCC ... 5V Prefer to use external 5V Source
  Gnd

  written By :Stelios Boultadakis,  03/03/2019
 */

//====== stepper motor declarations ======
#define IN1  8
#define IN2  9
#define IN3  10
#define IN4  11
int Steps = 0;
int Direction=1;
int Flag;
unsigned long last_time;
unsigned long currentMillis ;
float steps_left_per_rev=4095; //===== for motor BYJ48
float steps_left;
long time;
//======== serial control declarations ======
unsigned int integerValue=0;  // Max value is 65535
int i, len;
float result;
int rotations=0;
char incomingByte;
String directionString;
String rotationsString;
String incomingSerial="";  //  ' 'for one character and " " for multiple characters
//================================================
void setup() {
// ==== stepper motor control ====
pinMode(IN1, OUTPUT);
pinMode(IN2, OUTPUT);
pinMode(IN3, OUTPUT);
pinMode(IN4, OUTPUT);

// Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ;             // wait for serial port to connect. Needed for native USB port only
  }               // end of while
Serial.println("Hello from Arduino ...   F(for clockwise) or B(for counterclockwise) followed by the number of
rotations..  or S to Stop the program..");   //this command is necessary fro the installation of serial communication:

Serial.println();
}               // end of setup
//==================== Order for the stepper motor direction ======================
void SetDirection(){
if(Direction==1){ Steps++;}   //Clockwise rotation
if(Direction==0){ Steps--; }  //Counterclockwise rotation
if(Steps>7){Steps=0;}
if(Steps<0){Steps=7;}
```

```
}
//==== declare functions before loop because an error declare function was found during compilation ======
void stepper(int xw){
  for (int x=0;x<xw;x++){
    SetDirection();
switch(Steps){
   case 0:
     digitalWrite(IN1, LOW);
     digitalWrite(IN2, LOW);
     digitalWrite(IN3, LOW);
     digitalWrite(IN4, HIGH);
   break;
   case 1:
     digitalWrite(IN1, LOW);
     digitalWrite(IN2, LOW);
     digitalWrite(IN3, HIGH);
     digitalWrite(IN4, HIGH);
   break;
   case 2:
     digitalWrite(IN1, LOW);
     digitalWrite(IN2, LOW);
     digitalWrite(IN3, HIGH);
     digitalWrite(IN4, LOW);
   break;
   case 3:
     digitalWrite(IN1, LOW);
     digitalWrite(IN2, HIGH);
     digitalWrite(IN3, HIGH);
     digitalWrite(IN4, LOW);
   break;
   case 4:
     digitalWrite(IN1, LOW);
     digitalWrite(IN2, HIGH);
     digitalWrite(IN3, LOW);
     digitalWrite(IN4, LOW);
   break;
   case 5:
     digitalWrite(IN1, HIGH);
     digitalWrite(IN2, HIGH);
     digitalWrite(IN3, LOW);
     digitalWrite(IN4, LOW);
   break;
     case 6:
     digitalWrite(IN1, HIGH);
     digitalWrite(IN2, LOW);
     digitalWrite(IN3, LOW);
     digitalWrite(IN4, LOW);
   break;
   case 7:
     digitalWrite(IN1, HIGH);
     digitalWrite(IN2, LOW);
     digitalWrite(IN3, LOW);
     digitalWrite(IN4, HIGH);
   break;
   default:
     digitalWrite(IN1, LOW);
     digitalWrite(IN2, LOW);
     digitalWrite(IN3, LOW);
     digitalWrite(IN4, LOW);
   break;
```

```
}
}
}
//================================================================================
void loop() {
   Flag=1;
   if (Serial.available() > 0) {   // something came across serial
    incomingSerial= Serial.readString();// read the incoming data as string
    Serial.println( incomingSerial);   //******************
    if (!(incomingSerial.startsWith("F") || incomingSerial.startsWith("B")|| incomingSerial.startsWith("S") ))
      {
       Serial.println("WRONG message. TRY AGAIN ....");
       Flag=0; // αν ο πρώτος χαρακτήρας δεν είναι F, B ή S τότε η μεταβλητή rotations γίνεται 0
//         continue;
      }
      rotationsString=incomingSerial.substring(1);     // takes all characters after the first

     // convert rotations from String to Integer
     rotations=rotationsString.toInt();

   if (incomingSerial.startsWith("F")){
    Direction=0;
   }
   if (incomingSerial.startsWith("B")){
    Direction=1;
   }
   if (incomingSerial.startsWith("S")){
     Serial.println("Stops program...");
     return;
   }

   incomingSerial="";        //clears input String after reading completion

//========================== STEPPER MOTOR ROTATIONS =============
  steps_left=rotations*steps_left_per_rev*Flag;

//    Serial.println("steps_left");
//    Serial.println(round(steps_left));

while(steps_left>0){
  currentMillis = micros();
  if(currentMillis-last_time>=1000){
  stepper(1);
  time=time+micros()-last_time;
  last_time=micros();
  steps_left--;
  }
  }
 Serial.println("End of Rotation...");
}       // END IF Serial.available() > 0
//================================================================

}                       // end of loop
```

Στη παρακάτω εικόνα δίνεται ο κώδικας σε LabView:

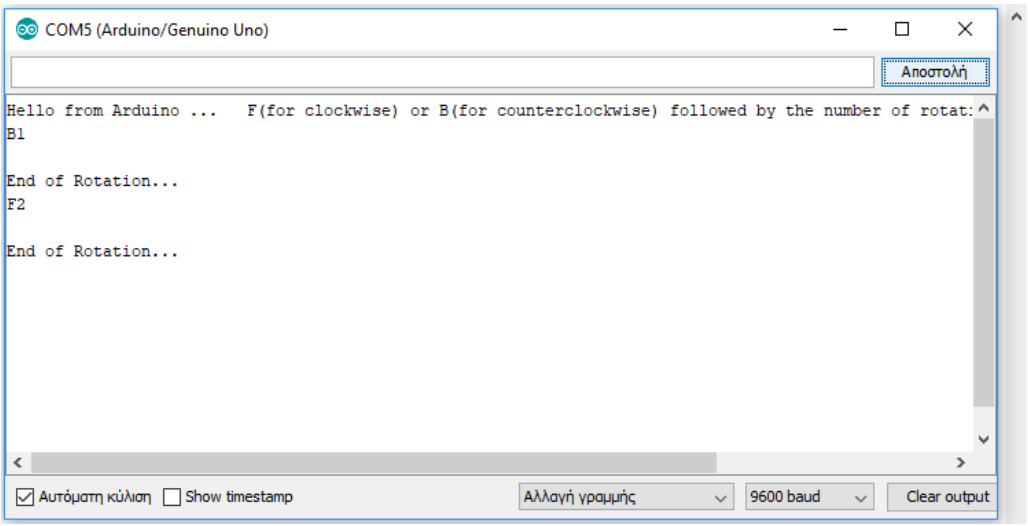




Προτείνεται επίσης να <u>οριστεί μια σταθερή και μεγάλη τιμή στην ιδιότητα **Device Timeout (100 sec)** ώστε να μην</u> εμφανίζεται μήνυμα σφάλματος όταν έχει δοθεί μεγάλος αριθμός στροφών στον κινητήρα οπότε λόγω της αργής ταχύτητας περιστροφής του, αργεί ο Arduino να αποστείλει μήνυμα ολοκλήρωσης στο LabView.

Για να αποσταλεί το μήνυμα της Εντολής Περιστροφής θα πρέπει αρχικά να έχουμε κάνει **ON** το αντίστοιχο κουμπί. Στις ρυθμίσεις της σειριακής θύρας στο LabView δηλώσαμε ότι το τέλος των μηνυμάτων δίνεται με την εντολή **'/n'** ή Dec(**10**) που σημαίνει τέλος γραμμής

Παρακάτω υπάρχει μια φωτογραφία της εφαρμογής <u>Παρακολούθησης Σειριακής Θύρας</u> (<u>Serial Monitor</u>) στο περιβάλλον προγραμματισμού του Arduino που φαίνονται τα μηνύματα που ανταλλάσσονται .



Προσοχή και στο Serial Monitor πρέπει να επιλέξουμε την '**Αλλαγή γραμμής'** καθώς και την ίδια ταχύτητα της σειριακής επικοινωνίας μεταξύ του Arduino και των εφαρμογών που τρέχουν κάθε φορά στον Η/Υ.

ΚΑΛΗ ΕΠΙΤΥΧΙΑ

<u>Αρχεία δοκιμών</u>

1) LabView_Comm_Arduino_Stepper-Motor_BYJ48_4_STELIOS.ino
2) Labview_Arduino_Serial_Comm_without Loop_5.vi