«Αξιοποίηση της ελεύθερης web εφαρμογής **tinkercad** στη διδασκαλία του μαθήματος της Ρομποτικής των ΕΠΑΛ. ».

Α) Εξοικείωση με το δικτυακό τόπο **tinkercad** της **AUTODESK**

- Θέτουμε σε ένα φυλλομετρητή τη διεύθυνση: https://www.tinkercad.com/
- Αποκτούμε πρόσβαση με έναν από τους διαθέσιμους τρόπους πατώντας **Sign In**
- Την πρώτη φορά δημιουργούμε Λογαριασμό Σύνδεσης επιλέγοντας **Join now** ή **Sign In/ New to Autodesk ? Create Account** οπότε συμπληρώνουμε τα στοιχεία μας στα πτυσσόμενα μενού που εμφανίζονται και δημιουργούμε λογαριασμό ο οποίος θα περιλαμβάνει:
  - Το πραγματικό e-mail μας για επικοινωνία και επιβεβαίωση/έλεγχο του λογαριασμού(π.χ stboultad@gmail.com)
  - Ένα password που θα ορίσουμε εμείς για να συνδεόμαστε στον ιστότοπο tinkercad και ο οποίος δεν είναι υποχρεωτικό να είναι ο πραγματικός κωδικός του gmail μας
- Με την ολοκλήρωση της δημιουργίας του λογαριασμού μεταφερόμαστε στο χώρο εργασίας που βλέπουμε και τη δυνατότητα δημιουργία 3D γραφικών και εκτυπώσεων σε 3D εκτυπωτή. Εμείς όμως δουλεύουμε με τις καρτέλες **Circuits.**
- Στην πάνω δεξιά μεριά της εφαρμογής βλέπουμε το εικονίδιο (πρόσωπο) του λογαριασμού μας και επιλέγουμε το σύνδεσμο **Learn** και στη συνέχεια στην αριστερή πλευρά αντί για στο πτυσσόμενο μενού **3D** αριστερά, επιλέγουμε διαδοχικά **Circuits/Projects/Show All Arduino**.
- Επιλέξτε ένα οποιοδήποτε project (προτείνουμε το Blink a LED with Digital Output) και στη συνέχεια επιλέξτε **Tinker This** ώστε να έρθετε στο περιβάλλον του εικονικού εργαστηρίου και της προσομοίωσης.
- Εξοικειωθείτε με τα μενού εργαλείων και τον τρόπο του σχεδιασμού του κυκλώματος και της προσομοίωσης ακολουθώντας τις οδηγίες από τα παράθυρο **Βοήθειας** που εμφανίζεται στο αριστερό τμήμα της εφαρμογής.
- Εξοικειωθείτε με τους ακροδέκτες του Arduino UNO (βλέποντας και την πραγματική πλακέτα) και τη δομή του προγράμματος όπως αυτό φαίνεται στα παράθυρα Βοήθειας.
- Εξοικειωθείτε με το παράθυρο κώδικα Code συζητώντας τον κώδικα επιλέγοντας τη μορφή κώδικα **Text** ώστε να είμαστε σε περιβάλλον Wiring C και στο Serial Monitor.
- Πατήστε το τετράγωνο κουμπί με τα πολύχρωμα γράμματα TINKERCAD στην αριστερή πλευρά της οθόνης και στην συνέχεια το κουμπί **Create New Circuit** για να σχεδιάσετε κυκλώματα. Κάθε φορά που δημιουργείτε ένα project είναι καλό να το μετονομάζετε κάνοντας διπλό κλικ πάνω στο τυχαίο όνομα που βάζει η εφαρμογή.
- Η Εφαρμογή αυτόματα αποθηκεύει όλες τις τελευταίες αλλαγές που κάνετε και κάθε φορά που θα εισέρχεστε στην καρτέλα **Circuits** βλέπετε τα κυκλώματα που έχετε δημιουργήσει και τα οποία μπορείτε να τα μοιραστείτε και με άλλους χρήστες.

- Για να προσομοιώσετε ένα προηγούμενο project, κάνετε κλικ πάνω του και στη συνέχεια επιλέγετε **tinker this**.
- Μπορείτε επίσης να μετονομάσετε ή να διαγράψετε ένα προηγούμενο project κάνοντας κλικ στο **πορτοκαλί γρανάζι** στο παραθυράκι του κάθε project.
- Κάθε φορά που θέλω να επιστρέψω στο παράθυρο με τα projects που έχω ήδη δημιουργήσει πατώ στο **τετράγωνο κουμπί με τα πολύχρωμα γράμματα TINKERCAD** στην αριστερή πλευρά της οθόνης .

Β) Βοήθεια για τη γλώσσα προγραμματισμού Wiring C και έτοιμα παραδείγματα κυκλωμάτων και κώδικα

- Δικτυακός Τόπος της Arduino. Ανοίξτε μια επιπλέον καρτέλα στο φυλλομετρητή σας και πληκτρολογήστε www.arduino.cc
- Κατεβάστε εάν δεν το έχετε ήδη, το περιβάλλον προγραμματισμού της Arduino
- Όλες οι εντολές, οι συναρτήσεις και οι βιβλιοθήκες επιλέγοντας **Resources/Reference** βλέποντας και τα προτεινόμενα παραδείγματα (https://www.arduino.cc/reference/en/)
- Επιλέξτε μέσα από τα έτοιμα παραδείγματα που υπάρχουν και στο περιβάλλον προγραμματισμού -διαδρομή **Resources/Tutorials/Built In Examples ή Examples from Libraries (**https://www.arduino.cc/en/Tutorial/HomePage) το Blink a Led και δείτε την πρόταση για τα απαραίτητα εξαρτήματα.
- Δείτε τους ακροδέκτες της πλακέτα Arduino Uno και συγκρίνετε με την πραγματική πλακέτα που έχετε μπροστά σας
- Δείτε το προτεινόμενο σχηματικό και τις επεξηγήσεις του κώδικα
- Πατήστε το κουμπί **Open Code** ώστε να μπορέσετε να αντιγράψετε τον κώδικα που επιθυμείτε
- Επικολλείστε ελεύθερα τον κώδικα είτε στο περιβάλλον προγραμματισμού του tinkercad για προσομοίωση είτε στο προγραμματιστικό περιβάλλον της Arduino ώστε να το κατεβάσετε στην πραγματική πλακέτα και στο κύκλωμα που φτιάξατε στο εργαστήριο

Γ) Προτεινόμενος τρόπος για το μάθημα στο σχολείο. Οι μαθητές:

- Δημιουργούν προσωπικούς λογαριασμούς στο tinkercad
- Ενημερώνονται από τον ιστότοπο της Arduino για τα απαραίτητα στοιχεία συγγραφής κώδικα στη Wiring C είτε αντιγράφουν ολόκληρο τον κώδικα
- Φορτώνουν σε επιπλέον καρτέλα του φυλλομετρητή το tinkercad (www.tinkercad.com), σχεδιάζουν το κύκλωμα και μεταφέρουν τον προηγούμενο κώδικα στο παράθυρο Code (Text).
- Κάνουν τις απαραίτητες τροποποιήσεις ενδεχομένως στους ακροδέκτες που θα χρησιμοποιήσουν και κάνουν έλεγχο σφαλμάτων πατώντας το κουμπί **Debugging** (εικονίδιο με τη **Μέλισσα**)

- Πατούν **Start Simulation** και ανοίγουν και το παράθυρο **Serial Monitor** εάν χρειάζεται.
- Φορτώνουν ταυτόχρονα στον Η/Υ το περιβάλλον προγραμματισμού της Arduino
- Αντιγράφουν τον κώδικα από το tinkercad στο περιβάλλον προγραμματισμού της Arduino
- Συνδέουν την πλακέτα Arduino Uno, επιλέγουν από το μενού **Εργαλεία/Πλακέτα** τη σωστή πλακέτα και ελέγχουν από τη διαδρομή **Εργαλεία/Θύρα** για τη αναγνώριση της πλακέτας Arduino Uno στη σωστή θύρα COM του Η/Υ. Για περισσότερες λεπτομέρειες ανατρέξτε στην ιστοσελίδα του 1ου ΕΠΑΛ Συκεών (http://1epal-sykeon.thess.sch.gr) στο σύνδεσμο **Εκπαιδευτικό Υλικό/Arduino** και στο φύλλο έργου: *Οδηγίες εγκατάστασης και χρήσης του Arduino Software (IDE)* …
- Εκτελούν **Debugging** και **μεταφόρτωση** του κώδικα μηχανής στην πλακέτα Arduino Uno και ελέγχουν τη λειτουργία του πραγματικού κυκλώματος.

Δ) Η Τεχνική PWM για την παραγωγή αναλογικού σήματος στις ψηφιακές εξόδους του ARDUINO UNO για έλεγχο DC motor (https://www.arduino.cc/en/Tutorial/PWM)

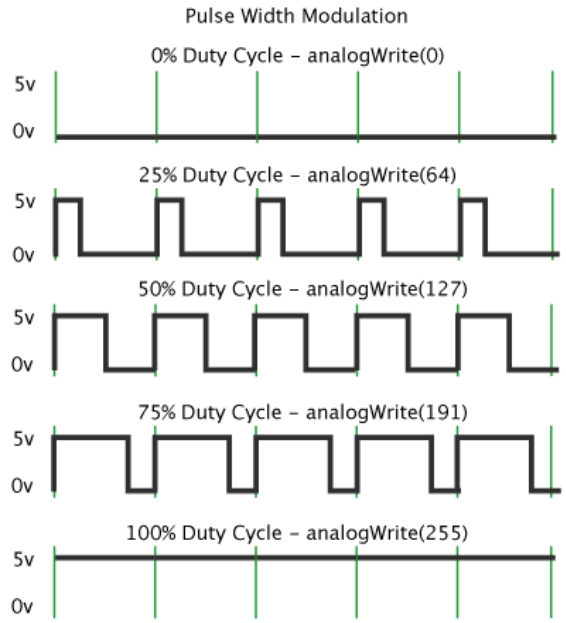
Με τη τεχνική **Pulse Width Modulation** (**PWM**) δημιουργούμε αναλογικά σήματα σε ακροδέκτες που λειτουργούν ως ψηφιακές έξοδοι. Στην πραγματικότητα μέσω της εντολής **analogWrite()** παράγουμε τετραγωνικούς παλμούς με διαφορετικούς κύκλους εργασίας (**duty cycles** %). Εφόσον η συχνότητα της τετραγωνικής κυματομορφής είναι αρκετά μεγάλη, μια συσκευή που συνδέεται σε ένα PWM pin αντιλαμβάνεται την κυματομορφή σαν μια DC τάση με τιμή ίσο με το μέσο όρο της τάσης της κυματομορφής. Η δομή της εντολής είναι:

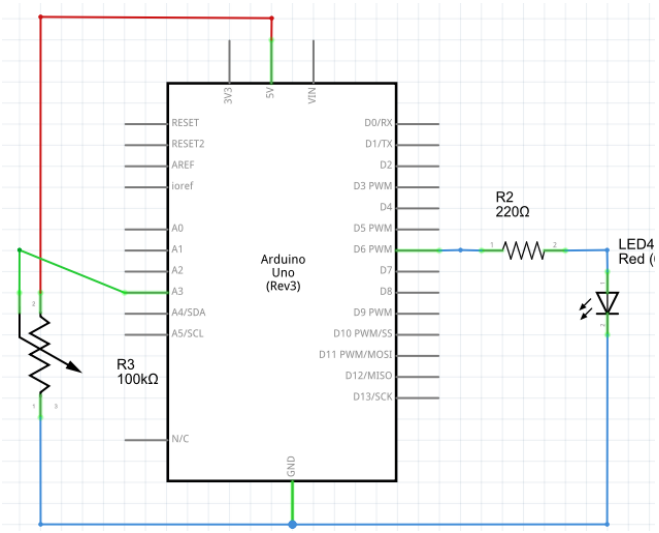
analogWrite(**ακροδέκτης**, περιεχόμενο **0-255**);

Όταν ο duty cycle=**0%** τότε το παραγόμενο σήμα έχει τιμή **0** Volts. Περιεχόμενο εντολής: **0**
Όταν ο duty cycle=**50**% τότε το παραγόμενο σήμα έχει τιμή **2.5** Volts. Περιεχόμενο εντολής: **127**
Όταν ο duty cycle=**100**% τότε το παραγόμενο σήμα έχει τιμή **5**Volts. Περιεχόμενο εντολής: **255**
Όταν ο duty cycle=**0-100**% τότε το παραγόμενο σήμα έχει τιμή **0-5**Volts. Περιεχόμενο : **0-255**

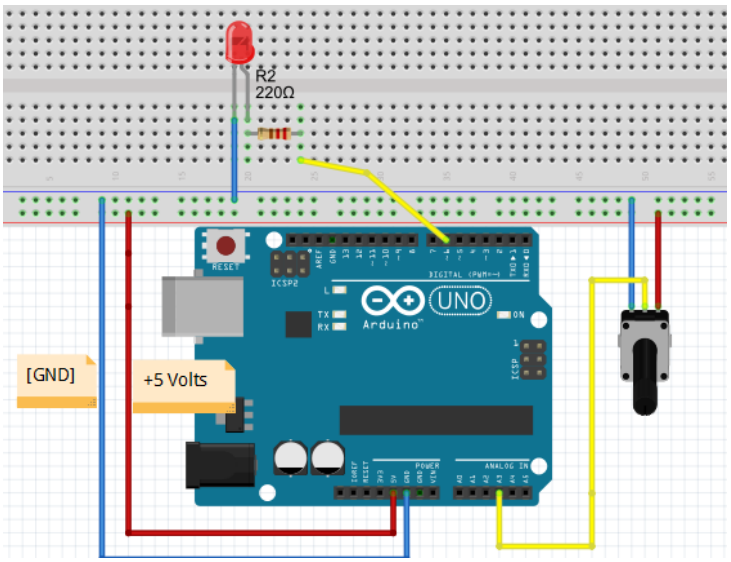
Η συχνότητα του τετραγωνικού παλμού για την πλακέτα μας είναι περίπου 500Hz δηλαδή έχει περίοδο 2ms οπότε για να προλαβαίνουμε να βλέπουμε το αποτέλεσμα της εντολής **analogWrite()** θα πρέπει να έχουμε ένα delay(ms)>2ms. Η εντολή **analogWrite()** δεν απαιτεί να έχουμε ορίσει ως OUTPUT

**** **ΠΡΟΣΕΞΤΕ:** οι αναλογικές είσοδοι (A0-A5) που διαβάζονται με την εντολή analogRead(…) προφανώς δεν ορίζονται ως είσοδοι με εντολές pinmode. Το ίδιο ισχύει και για τους ψηφιακούς ακροδέκτες που έχουν την ένδειξη με το σύμβολο ~ όταν χρησιμοποιούνται ως **έξοδοι PWM με την εντολή** analogWrite(..,..) (δεν χρειάζεται να τους ορίσουμε ως εξόδους με εντολές pinmode).*





*<u>Τα σχήματα έγιναν με το **Fritzing**</u>*



Είσοδος: μια μεταβαλλόμενη αναλογική τάση από το ποτενσιόμετρο στην αναλογική είσοδο **A3** του Arduino UNO που αντιστοιχεί σε επίπεδα κβάντισης 0-1023 (0-5 Volts) με την εντολή:

val = analogRead(analogPin);

και στέλνουμε σε **έξοδο PWM** (ψηφιακές έξοδοι με το σύμβολο **~**) που στην περίπτωσή μας είναι ο ψηφιακός ακροδέκτης **6**, ένα τετραγωνικό παλμό με διάρκεια 0-255 (0-5 Volts) με την εντολή:

analogWrite(ledPin, val / 4);

<u>Γράφουμε τον παρακάτω κώδικα:</u>

```
int ledPin = 6;                 // ορίζουμε σαν ψηφιακή έξοδο για το LED το pin 9
int analogPin = 3;              // ορίζουμε σαν αναλογική είσοδο για το ποτενσιόμετρο το pin 3
int val = 0;                    // ορίζουμε ως ακέραια τη μεταβλητή εισόδου με όνομα val
void setup()
{
//  pinMode(ledPin, OUTPUT);      // οι ψηφιακοί ακροδέκτες PWM δεν χρειάζονται να
}                                 //οριστούν ούτε ως είσοδοι ούτε ως έξοδοι ***
void loop()
{
  val = analogRead(analogPin);   // read the input pin
  analogWrite(ledPin, val / 4);  // analogRead values go from 0 to 1023,
                                    //analogWrite values from 0 to 255
}
```

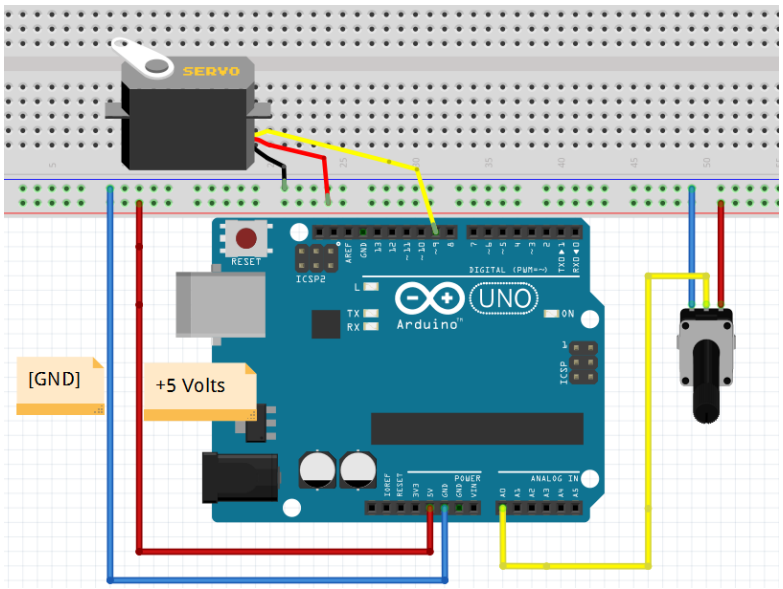
Ε) Έλεγχος SERVO MOTOR με την κλάση Servo της Arduino

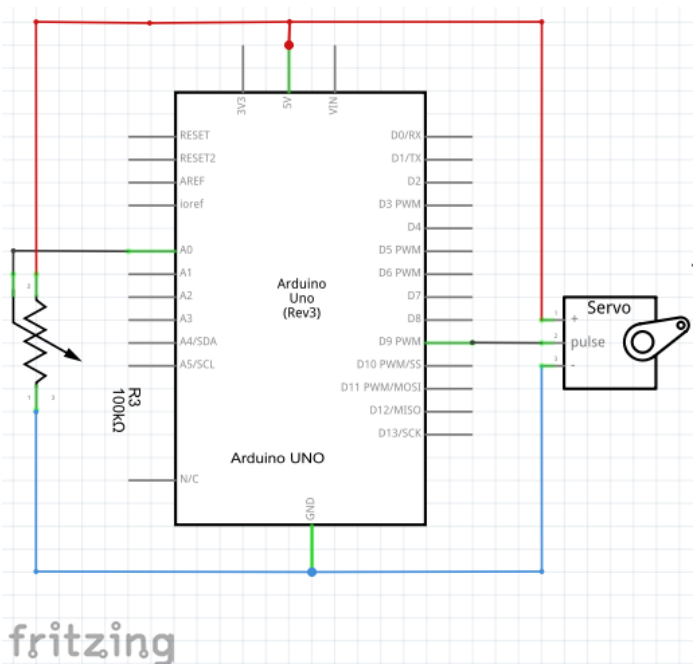
Με ένα ποτενσιόμετρο δημιουργούμε μια μεταβαλλόμενη αναλογική τάση η οποία διαβάζεται από το μικροελεγκτή. Στη συνέχεια ο μικροελεγκτής δημιουργεί ένα σήμα PWM με το οποίο καθορίζεται η γωνία στροφής του servo motor

Η βιβλιοθήκη Servo μας βοηθάει στον προγραμματισμό του ελέγχου ενός σερβοκινητήρα (Servo Motor) όπως εκείνους που χρησιμοποιούνται στο ραδιομοντελισμό (RC-Hobby servo motor) από την πλακέτα Arduino. Η ενσωμάτωση στον κώδικα της βιβλιοθήκης Servo γίνεται με την εντολή #include ή μέσω της διαδρομής **Σχέδιο(Sketch)/Συμπερίληψη Βιβλιοθήκης**. Στη συνέχεια γράφουμε κώδικα αντίστοιχο με εκείνο της Κλάσης / Αντικειμένων Κλάσης στη γλώσσα C++. Έτσι στον κώδικα που θα γράψουμε δεν θα κάνουμε χρήση των εντολών **pinmode(pin, input/outpout)** και **analogwrite(pin, value)** αλλά θα χρησιμοποιήσουμε ενσωματωμένες συναρτήσεις - εντολές της βιβλιοθήκης Servo που τις βρίσκουμε στη διεύθυνση https://www.arduino.cc/en/Reference/Servo

Ο σερβοκινητήρας διαθέτει σύστημα από γρανάζια (integrated gears) και έναν άξονα (shaft) που μπορεί να ελεγχθεί με ακρίβεια. Τα πιο συνηθισμένα servo motors επιτρέπουν ο άξονας να τοποθετείται σε διάφορες γωνίες, συνήθως μεταξύ 0 και 180 μοιρών. Το servo motor που χρησιμοποιούμε έχει τρείς ακροδέκτες:

- **Κόκκινο** :για **+5 Volts**
- Μαύρο : **για γείωση**
- **Πορτοκαλί** : για σύνδεση του σήματος **PWM** από έναν αντίστοιχο ψηφιακό ακροδέκτη της πλακέτας Arduino UNO με την ένδειξη **~** (** *Το σχέδιο έγινε με το πρόγραμμα Fritzing)*





Ανάλυση του κώδικα που θα γράψουμε:

1) Διαβάζουμε μια μεταβαλλόμενη αναλογική τάση στην αναλογική είσοδο **A0** του Arduino UNO από το ποτενσιόμετρο που αντιστοιχεί σε επίπεδα κβάντισης 0-1023 (0-5 Volts) με την εντολή *val = analogRead(analogPin);*

2) Μετατρέπουμε την τιμή *val* με εύρος τιμών από *0-1023* σε τιμές με εύρος τιμών *0-255* με την εντολή *val = map(val, 0, 1023, 0, 180)* ; ώστε να είναι συμβατή με τις τιμές με τις οποίες λειτουργεί το servo motor.

3) Στέλνουμε στην έξοδο PWM (ψηφιακές έξοδοι με το σύμβολο **~**) που στην περίπτωσή μας είναι ο ψηφιακός ακροδέκτης **9**, ένα τετραγωνικό παλμό με διάρκεια 0-180 (0-5 Volts) με την εντολή *myservo.write(val) ;*.

Για να θυμηθούμε τι είναι το σήμα PWM μπορούμε να ανατρέξουμε στο Φύλλο Έργου:
«*ΦΩΤΑ ΟΔΟΣΗΜΑΝΣΗΣ ΤΕΧΝΙΚΩΝ ΕΡΓΩΝ (αναλογική έξοδος FADING).doc*»

Γράφουμε τον παρακάτω κώδικα:

```
#include <Servo.h>// φορτώνουμε τη βιβλιοθήκη Servo για τον
                  // έλεγχο σερβοκινητήρα servo motor

Servo srvmotor1; //δημιουργούμε ένα αντικείμενο της κλάσης Servo με το
                  // όνομα  srvmotor1.Από εδώ και πέρα το αντικείμενο
                  // srvmotor1 θα λειτουργεί με τις ίδιες εντολές που
                  // συμπεριλαμβάνονται στη βιβλιοθήκη κλάση Servo. Αν έχω
                  // και δεύτερο σερβοκινητήρα δημιουργώ και δεύτερο
                  // αντικείμενο srvmotor2με την εντολή Servo srvmotor2;
                  // και στη συνέχεια αντιγράφω τις ίδιες εντολές ελέγχου
                  // για το αντικείμενο srvmotor2

int potpin = 0;       // ορίζω σε ποια αναλογική είσοδο(pin) θα συνδέσω
                  // το potentiometer, δηλαδή την είσοδο A0 του ArduinoUno

int val;              // ορίζω τη μεταβλητή val που θα παίρνει τις τιμές
                  // από το potentiometer. Οι τιμές αυτές θα έχουν εύρος
                  // από0-1023 λόγω του 10 bit A/D μετατροπέα της
                  // αναλογικής τάσης που εφαρμόζεται στον αναλογικό
                  // ακροδέκτη A0

void setup() {
 srvmotor1.attach(9);  // με την εντολή attach της κλάσης Servo
                  // δηλώνω ότι το αντικείμενο srvmotor1 που αντιστοιχεί
                  // στο σερβοκινητήρα μου θα παίρνει σήμα από τον
                  // ακροδέκτη 9 της πλακέτας ArduinoUno. Συνδέω δηλαδή
                  // το πορτοκαλί καλώδιο του σερβοκινητήρα στην υποδοχή
                  // του ArduinoUno ~9
 }

void loop() {
  val = analogRead(potpin); // διαβάζω την αναλογική τάση από το potentiometer
                          // που είναι σε κλίμακα από 0 έως 1023 λόγω A/D

   val = map(val, 0, 1023, 0, 180); // αλλάζω την κλίμακα της μεταβλητής val
                                // από 0-1023 σε κλίμακα 0-180 στην οποία
                                //είναι η γωνία περιστροφής του servo motor
```

```
  srvmotor1.write(val);  // με την εντολή write της κλάσης Servo στρέφω το
                         // αντικείμενο srvmotor1 (δηλαδή servo motor) σε
                         // γωνίες 0-180

  delay(15);            // καθυστέρηση 15 ms ώστε να φτάσει στη θέση του
                        // το servo motor
}
```

ΣΤ) Έλεγχος STEPPER MOTOR με τις κλάσεις της Arduino

Βήμα 1ο: Τι πρέπει να γνωρίζουμε:

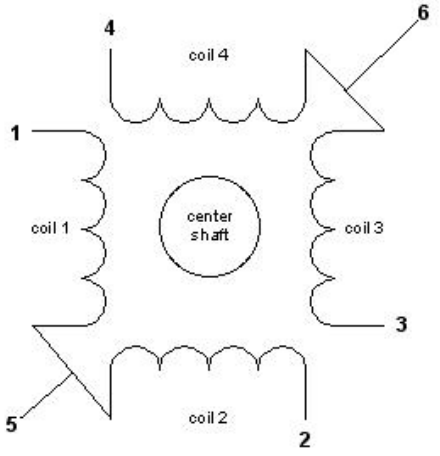
Οι βηματικοί κινητήρες διαφέρουν από τους άλλους τύπους κινητήρων συνεχούς και εναλλασσομένου ρεύματος στο ότι τροφοδοτούνται με παλμούς και παράγουν ηλεκτρική κίνηση. Ο άξονας τους δεν έχει μια συνεχή περιστροφική κίνηση, αλλά περιστρέφεται κατά μία γωνία κάθε φορά που δέχεται ένα παλμό. Ένας βηματικός 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.

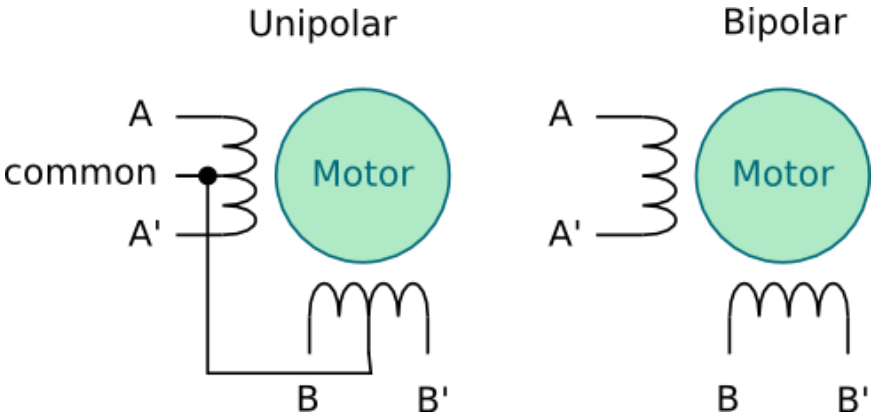
Μονοπολικοί βηματικοί κινητήρες

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:

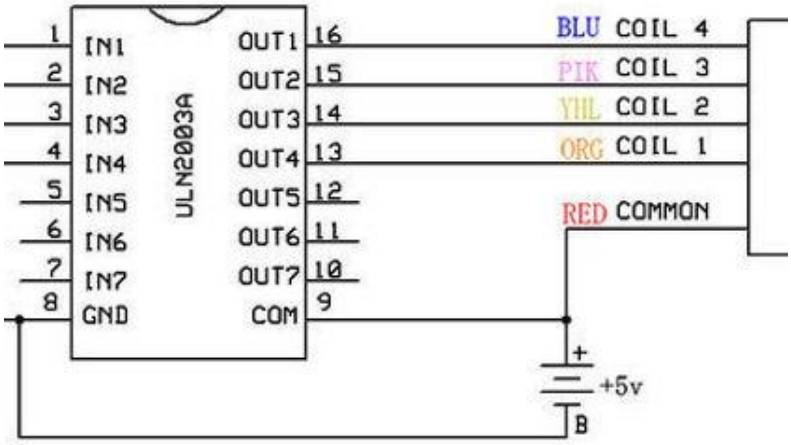
Τύπος κινητήρα: Unipolar Stepper Motor 28-BYJ48 που συνοδεύεται από την πλακέτα οδηγού

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

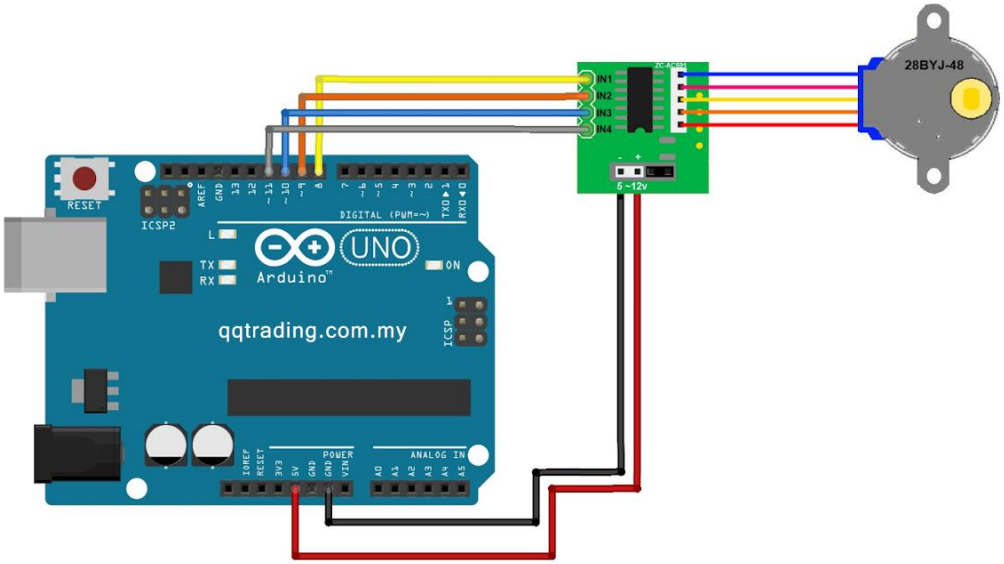
Το μοτέρ έχει ένα λόγο γραναζιών : Gear ratio=64

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

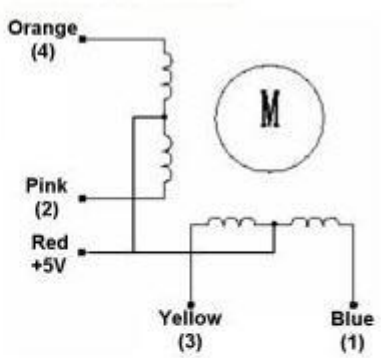
ενώ η γωνία που κάνει σε κάθε βήμα (Stride Angle)= 7.5°

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=πολλοί χαρακτήρες) 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();
```

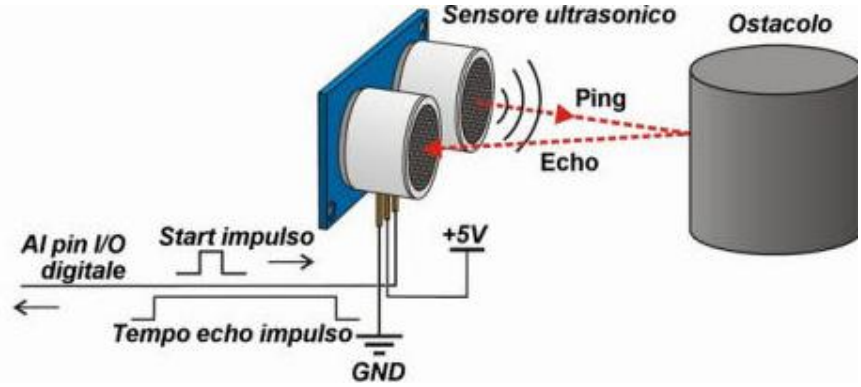
Μετατροπή ενός αλφαριθμητικού δεδομένου σε ακέραιο αριθμό με βάση τις ιδιότητες του πίνακα ASCII:

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

Ζ) ΛΕΙΤΟΥΡΓΙΑ ΤΟΥ ΑΙΣΘΗΤΗΡΑ ΑΠΟΣΤΑΣΗΣ HC-SR04

Τι πρέπει να γνωρίζουμε:

Ο αισθητήρας HC-SR04 χρησιμοποιείται για μετρήσεις αποστάσεων. Όπως φαίνεται στην παρακάτω εικόνα έχει δύο ‘μάτια’ που στην πραγματικότητα είναι ένας πομπός που στέλνει υπερήχους προς τα γύρω αντικείμενα και ένας δέκτης υπερήχων που δέχεται τους υπερήχους μετά την ανάκλασή τους στα αντικείμενα που βρίσκει μέσα στο ‘οπτικό’ του πεδίο. Μπορεί να μετρήσει αποστάσεις από 2cm μέχρι 400cm και με ακρίβεια 0,3cm.



https://tuixte.wordpress.com/2013/03/05/how-to-ultrasonic-sensor-hc-sr04/

Πως λειτουργεί ο αισθητήρας

Περιλαμβάνει τέσσερις ακροδέκτες από τους οποίους :

- οι δύο εξωτερικοί είναι για την τροφοδοσία του **Vcc** (5 Volts) και **GND** που μπορούν να τροφοδοτηθούν από την πλακέτα Arduino UNO. Από τους δύο κεντρικούς ακροδέκτες
- ο **Trig** είναι ακροδέκτης εισόδου στον αισθητήρα (άρα θα συνδεθεί σε ακροδέκτη εξόδου του Arduino UNO) και χρησιμεύει για την ενεργοποίηση (σκανδαλισμό, triggering). Θα πρέπει να λάβει από το Arduino UNO έναν παλμό που θα έχει διάρκεια του μετώπου HIGH τουλάχιστον 10μs (microseconds).
- ο **Echo** είναι ακροδέκτης εξόδου του αισθητήρα (άρα θα συνδεθεί σε ακροδέκτη εισόδου του Arduino UNO) και στον οποίο όταν φτάνει ο ανακλώμενος από το αντικείμενο υπέρηχος , παράγει ένα παλμό HIGH του οποίου η διάρκεια σε microseconds ταυτίζεται με το χρόνο που μεσολαβεί από τη στιγμή που έφυγε ο υπέρηχος από τον σκανδαλισμό μέχρι να γυρίσει πίσω στον ακροδέκτη Echo. Στην πραγματικότητα μετράει δύο φορές την απόσταση (μια για να πάει και μια για να γυρίσει).

Πως παράγεται ο παλμός triggering

Τα βήματα που ακολουθούμε είναι τα παρακάτω βήματα:

- βάζουμε τον ακροδέκτη **Trig** σε κατάσταση LOW για 2μs

```
digitalWrite(trig, LOW);
delayMicroseconds(2);
```

- βάζουμε τον ακροδέκτη **Trig** σε κατάσταση HIGH για 10μs

```
digitalWrite(trig, HIGH);
delayMicroseconds(10);
```

- βάζουμε τον ακροδέκτη **Trig** σε κατάσταση LOW για 2μs

```
digitalWrite(trig, LOW);
delayMicroseconds(2);
```

Πως διαβάζεται ο ανακλώμενος υπέρηχος στον ακροδέκτη **Echo** και υπολογίζεται η απόσταση

Χρησιμοποιούμε την εσωτερική συνάρτηση **pulseIn(ακροδέκτης, HIGH/LOW)** της γλώσσας Wiring C.βιβλιοήκη. Η συνάρτηση μετράει το χρόνο ενός μετώπου (HIGH ή LOW) ενός παλμού που έρχεται σε ένα δηλωμένο ακροδέκτη. Με την παρακάτω εντολή μετράμε τη διάρκεια ενός παλμού HIGH που έρχεται στον ακροδέκτη Echo και τον αποδίδουμε στη μεταβλητή duration σε μονάδες microseconds.

```
duration = pulseIn(echo, HIGH);
```

Στη συνέχεια παίρνουμε υπόψη τον τύπο της ευθύγραμμης ομαλής κίνησης **s=u*t** όπου **u** είναι η ταχύτητα του ήχου. Είναι γνωστό πως **u=340m/s** και αν μετατρέψουμε τα μέτρα σε cm (*100) και το χρόνο σε microseconds (*10^6) τότε η ταχύτητα του ήχου γίνεται:

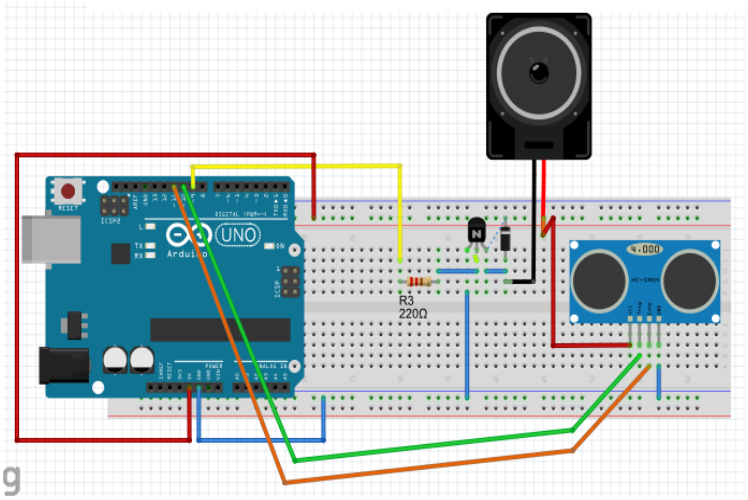
u=340*10^2/10^6=340/10000=1/29 cm/μς

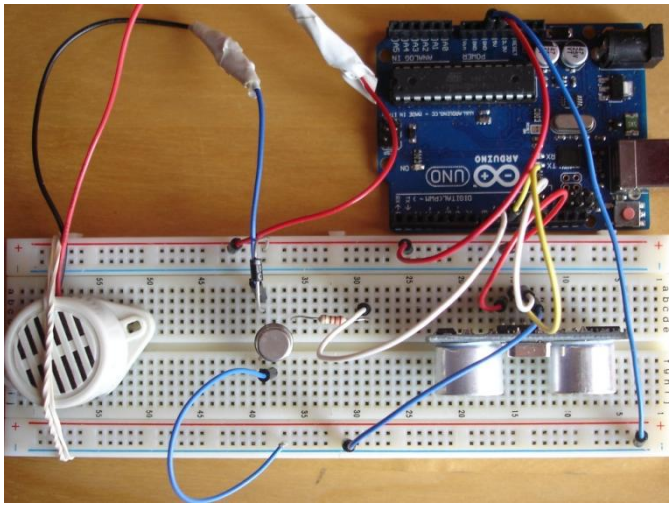
οπότε η υπολογιζόμενη απόσταση είναι **s=t*u (χρόνος * ταχύτητα του ήχου)**:

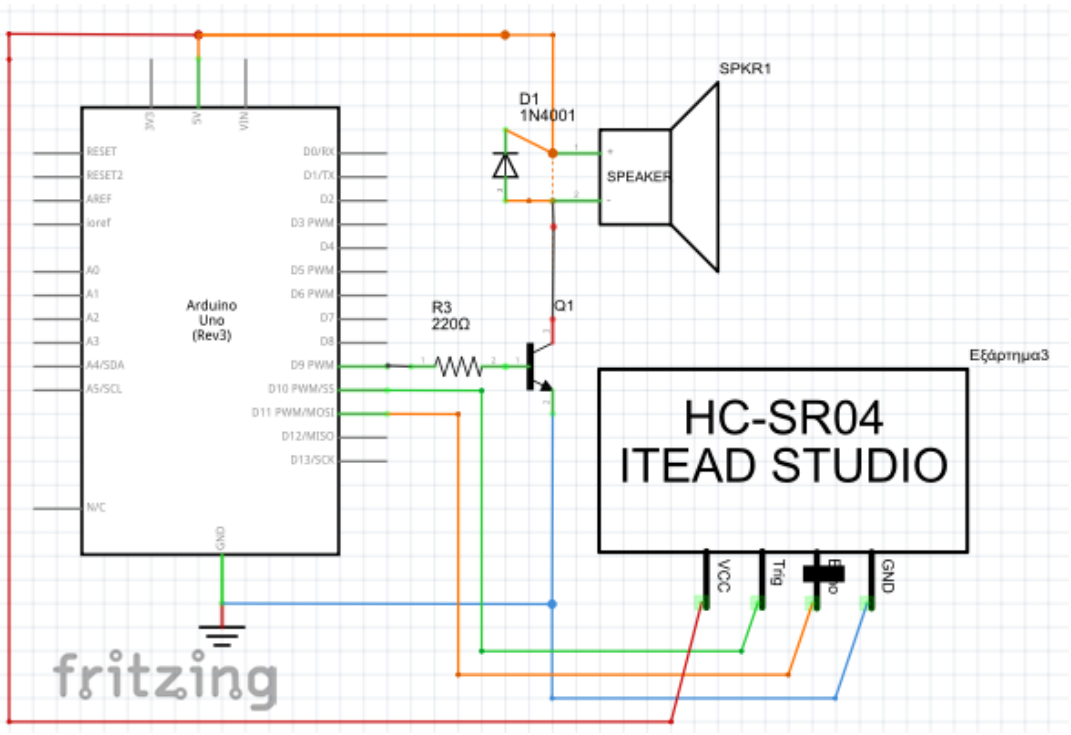
```
cm=duration / 29 /2;
```

Η επιπλέον διαίρεση με το 2 προκύπτει γιατί μετράμε δύο φορές την απόσταση (μια για να πάει και μια για να γυρίσει ο υπέρηχος) όπως αναφέραμε και παραπάνω.

Το κύκλωμα της εφαρμογής μας φαίνεται στις παρακάτω εικόνες:







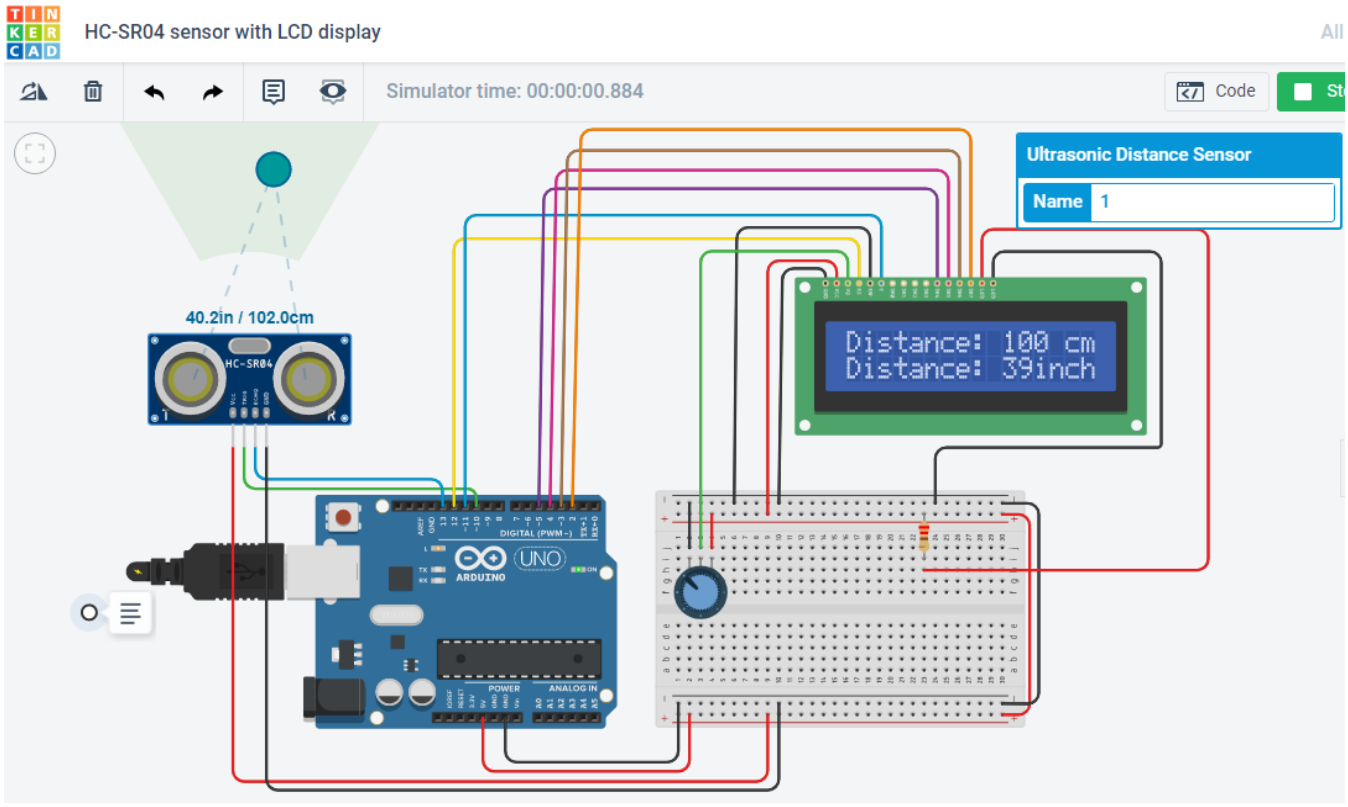
Ένας ενδεικτικός κώδικας σε γλώσσα Wiring είναι ο παρακάτω:

```
int trig = 10;

int echo = 11;

int buzz=9;

long duration, cm;


void setup()
```

```
{
  pinMode(trig,OUTPUT);
  pinMode(echo,INPUT);
  pinMode(buzz, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  //δημιουργία παλμού triggering
  digitalWrite(trig, LOW);
  delayMicroseconds(2);
  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig, LOW);
  delayMicroseconds(2);

  // μέτρηση της διάρκειας του μετώπου HIGH στον παλμό ECHO
  duration = pulseIn(echo, HIGH);

 //υπολογισμός της απόστασης σε cm και αποστολή της μέτρησης στην οθόνη
  cm=duration / 29 /2;
  Serial.println(cm);

//παραγωγή ήχου: ενεργοποίηση του βομβητή
  beep(cm);
  delay(50);    // είναι απαραίτητο για να προλαβαίνει να γίνεται η εκτίμηση της απόστασης
  }
```

```
//με μεταλλικό αντικείμενο να κάνω ανακλάσεις και με το χέρι
// ηχητικός εντοπισμός αντικειμένου

void beep(long cm){
// στη συνάρτηση αυτή παράγω ήχο στον οποίο το LOW μέτωπο είναι αντίστοιχο της
// απόστασης. Δηλαδή μεγάλη απόσταση δίνει αργό ρυθμό ήχου ενώ μικρή απόσταση δίνει //γρήγορο ρυθμόήχου
  long del=0;
  del=10*cm;
  if (cm < 120)                    //ΑΝ ΠΛΗΣΙΑΖΕΙ ΕΝΑ ΑΝΤΙΚΕΙΜΕΝΟ:
  {
  digitalWrite(buzz, HIGH);        // ON buzzer
  delay(200);                      // 400 ms ON
  digitalWrite(buzz, LOW);         // OFF buzzer
  delay(del);                      // stay OFF αντίστοιχα με την απόσταση
  }
  else
  {
  digitalWrite(buzz, LOW);      // ΑΝ ΤΟ ΑΝΤΙΚΕΙΜΕΝΟ ΕΙΝΑΙ ΜΑΚΡΙΑ:ΟΧΙ ΗΧΟ
  }
}
```

Η) Μετρητής Απόστασης και Απεικόνιση σε οθόνη LCD1602

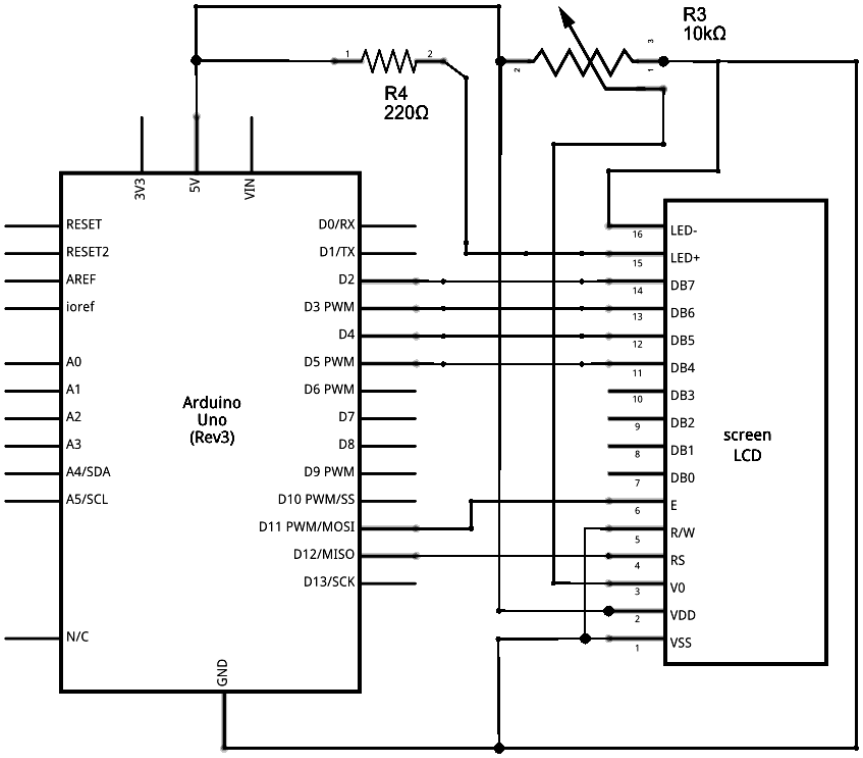


Πως συνδέεται η οθόνη LCD με το Arduino:

Για να ελέγξουμε την οθόνη LCD πρέπει να συνδέσουμε κατάλληλα τους ακροδέκτες της με τους ακροδέκτες του Arduino. Συγκεκριμένα:

LDC Pins **Arduino Pins**

- VSS:Συνδέεται στη γείωση του Arduino (pin GND)
- VDD:Συνδέεται στο pin 5V του Arduino
- V0 Contrast: Συνδέεται σε ποτενσιόμετρο
- RS: Συνδέεται σε ένα ψηφιακό pin του Arduino (D12)
- RW:Συνδέεται στη γείωση για να διαβάσουμε από τους καταχωρητές της οθόνης
- E: Συνδέεται σε ένα ψηφιακό pin του Arduino (D11)
- DB0-DB3: Μένουν ασύνδετοι
- DB4-DB7: Συνδέονται σε 4 ψηφιακά pin του Arduino (D5-D4-D3-D2)
- Anode: Συνδέεται στα 5V μέσω αντίστασης 220 Ω
- Kathode: Συνδέεται στη γείωση



Οδηγίες για την ενσωμάτωση βιβλιοθήκης στο Arduino

Για να ενσωματώσουμε μία βιβλιοθήκη που περιέχει οδηγίες για τη λειτουργία του αισθητηρίου που θα χρειαστούμε κάνουμε τις εξής ενέργειες:

1. Πρώτα κατεβάζουμε τη βιβλιοθήκη στον υπολογιστή μας σε μορφή ZIP **(DHTLib)** από κάποια σχετική ιστοσελίδα, π.χ.

   http://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-an-arduino/

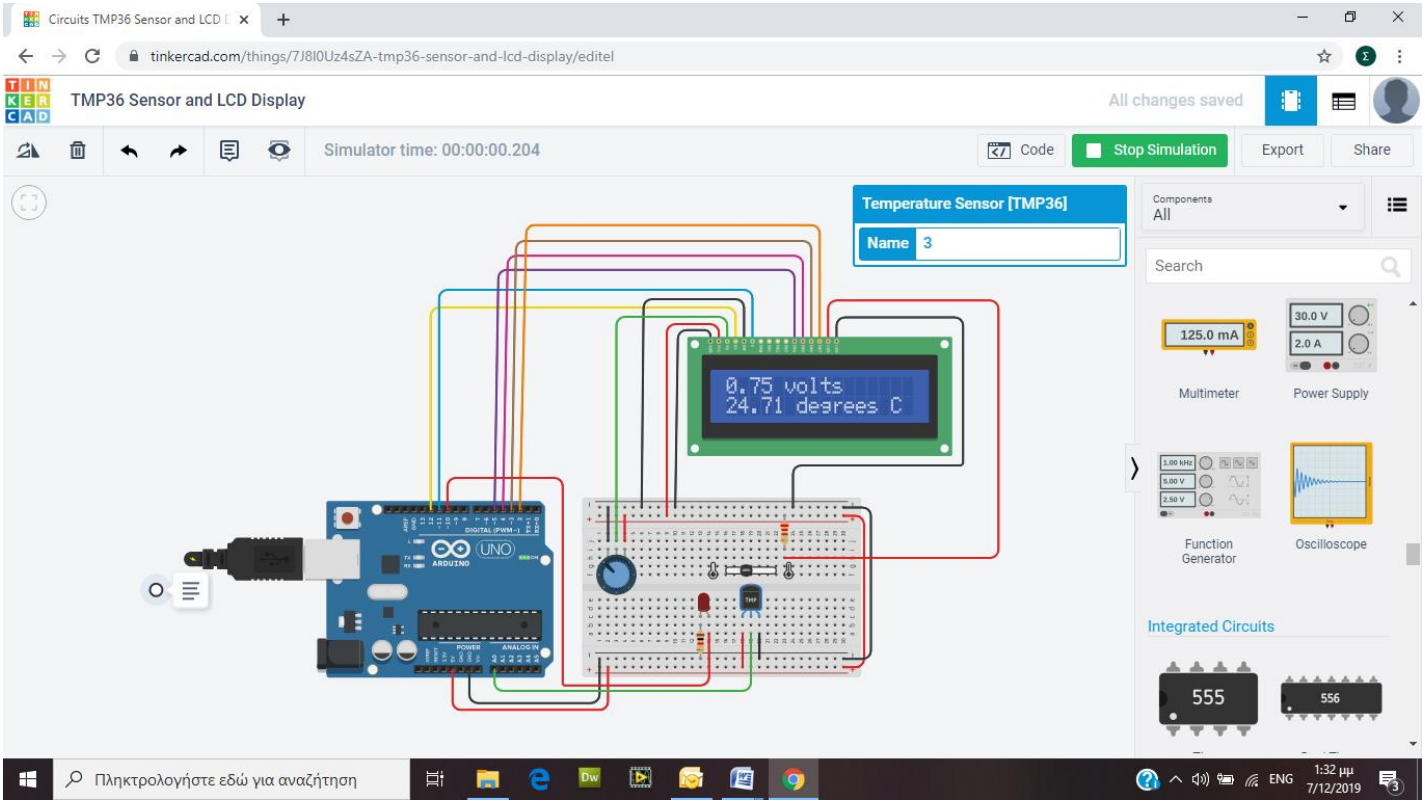
2. Από το περιβάλλον IDE του Arduino και το μενού επιλογών **Σχέδιο** επιλέγουμε **Συμπερίληψη βιβλιοθήκης** και στη συνέχεια **Προσθήκη βιβλιοθήκης ZIP…**
3. Επιλέγουμε το συμπιεσμένο αρχείο της βιβλιοθήκης ή το φάκελο που περιέχει τη βιβλιοθήκη και πατάμε το κουμπί **Open.** Η βιβλιοθήκη έχει ενσωματωθεί στο περιβάλλον εργασίας του Arduino και είναι έτοιμη για χρήση.

Ο Κώδικας

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12 , 11, 5, 4, 3, 2);

const int trigPin = 10;

const int echoPin = 13;
```

```
long duration;
int distanceCm, distanceInch;
void setup() {
        lcd.begin(16,2);
        pinMode(trigPin, OUTPUT);
        pinMode(echoPin, INPUT);
}
void loop() {
        digitalWrite(trigPin, LOW);
        delayMicroseconds(2);
        digitalWrite(trigPin, HIGH);
        delayMicroseconds(10);
        digitalWrite(trigPin, LOW);
        duration = pulseIn(echoPin, HIGH);
        distanceCm= duration*0.034/2;
        distanceInch = duration*0.0133/2;
        lcd.setCursor(0,0);
        lcd.print("Distance: ");
        lcd.print(distanceCm);
        lcd.print(" cm");
        delay(10);
        lcd.setCursor(0,1);
        lcd.print("Distance: ");
        lcd.print(distanceInch);
        lcd.print("inch");
        delay(10);
}
```

Η) Μετρητής Θερμοκρασίας με το TMP36 (αντίστοιχο του LM35) και Απεικόνιση σε οθόνη LCD1602




```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12 , 11, 5, 4, 3, 2);

const int LM35_Pin = A0;

const int LED_Pin=10;

const float temp_limit=40.00;

float temperature=0.00;

int temper=0;

void setup() {

 lcd.begin(16,2);

 pinMode(LED_Pin, OUTPUT);

 Serial.begin(9600);

}

void loop() {

 int reading = analogRead(LM35_Pin);
```

```
float voltage = reading * 5.0;
voltage /= 1024.0;
Serial.print(voltage); Serial.println(" volts");
float temperatureC = (voltage - 0.5) * 100 ;
Serial.print(temperatureC);
Serial.println(" degrees C");
float temperatureF = (temperatureC * 9.0 / 5.0) + 32.0;
Serial.print(temperatureF); Serial.println(" degrees F");
if (temperatureC>temp_limit)
{
 digitalWrite(LED_Pin, HIGH);
}
 else
{
 digitalWrite(LED_Pin, LOW);
}
 lcd.setCursor(0,0);
lcd.print(voltage);
lcd.print(" volts");
delay(10);
lcd.setCursor(0,1);
lcd.print(temperatureC);
lcd.print(" degrees C");
delay(10);
}
```