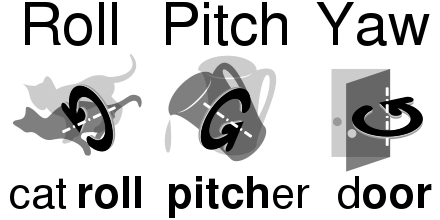
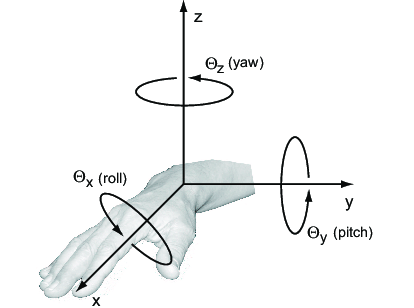
Σκοπός αυτό του project είναι η κατασκευή και ο προγραμματισμός ενός ρομποτικού βραχίονα τριών βαθμών ελευθερίας. Όταν λέμε βαθμός ελευθερίας μιλάμε για το πόσες πληροφορίες θέλουμε για να καθορίσουμε ακριβώς το θέση του βραχίονα στο χώρο. Για παράδειγμα σε ένα επίπεδο, για να ξέρουμε ακριβώς που είναι ένα αντικείμενο χρειαζόμαστε:

1. Τις συντεταγμένες του x και y
2. Αλλά και τον προσανατολισμό του, δηλαδή τη γωνία προς την οποία «κοιτάει» το αντικείμενο.



Στον τρισδιάστατο χώρο οι βαθμοί ελευθερίας ενός αντικειμένου είναι **εξι!** Τρεις συντεταγμένες και τρεις γωνίες!

Για τον χειρισμό του βραχίονα έχουμε τρεις επιλογές:

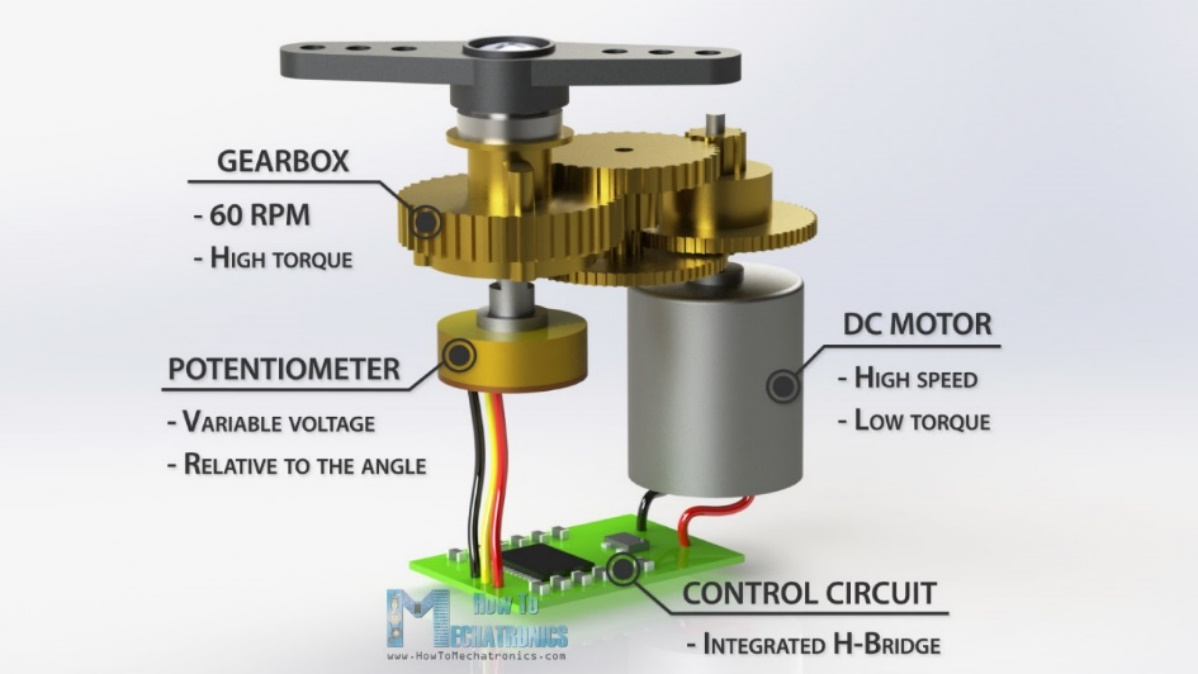
1. Ενσύρματο τηλεχειριστήριο το οποίο θα φτιάξουμε
2. Μέσω Bluetooth
   1. Κινητό
   2. Ασύρματο τηλεχειριστήριο

A picture containing text

Description automatically generatedΤα στοιχεία που μπορούν να αλληλοεπιδράσουν με το φυσικό κόσμο μέσω κίνησης, τα ονομάζουμε επενεργητές (actuators).

Ένα από αυτό είναι ο σερβοκινητήρας.

Ο συγκεκριμένος έχει τεράστιο εύρος χρήσης καθώς μας επιτρέπει να ξέρουμε ανά πάσα στιγμή τη θέση στην οποία βρίσκεται. Είναι ένας επενεργητής με έλεγχο θέσης! Το εσωτερικό ενός σερβοκινητήρα φαίνεται στη παρακάτω εικόνα.



Πολύ σημαντικό κατά τη χρήση σερβοκινητήρων είναι να δούμε τα τεχνικά χαρακτηριστικά τους:

**Specifications:**

**Working voltage: DC 4.8V ~ 6V**

**Operating angle range: about 180 ° (at 500 → 2500 μsec)**

Pulse width range: 500 → 2500μsec

No-load speed: 0.12 ± 0.01 sec / 60 (DC 4.8V) 0.1 ± 0.01 sec / 60 (DC 6V)

**No-load current: 200 ± 20mA (DC 4.8V)220 ± 20mA (DC 6V)**

Stopping torque: 1.3 ± 0.01kg · cm (DC 4.8V) 1.5 ± 0.1kg · cm (DC 6V)

Stop current: ≦ 850mA (DC4.8V) ≦ 1000mA (DC 6V)

**Standby current: 3 ± 1mA (DC 4.8V) 4 ± 1mA (DC 6V)**

Lead length: 250 ± 5 mmAppearancesize: 22.9 \* 12.2 \* 30mmWeight: 9 ± 1 g (without servo horn)

**Storage temperature: -20 ℃ ~ 60 ℃**

**Operating temperature:-10 ℃ ~ 50 ℃**

**Συνδεσμολογία (1 servo)**

Graphical user interface

Description automatically generated

Για να προγραμματίσουμε έναν σερβοκινητήρα θα πρέπει να του δώσουμε σήμα PWM, δηλαδή πρέπει να το συνδέσουμε στις θύρες που υποστηρίζουν PWM (~).

Για την κίνηση των συνδέσμων θα χρησιμοποιήσουμε τέσσερις σερβοκινητήρες.

|  |  |  |
| --- | --- | --- |
| **Όνομα** | **0°** | **180°** |
| Servo 1（Βάση） | Τέρμα δεξιά | Τέρμα αριστερά |
| Servo 2（Δεξιά πλευρά） | Επέκταση | Επιστροφή |
| Servo 3（Αριστερή πλευρα） | Επιστροφή | Επέκταση |
| Servo 4（Δαγκάνα) | Κλειστή | Ανοικτή |

Για το προγραμματισμό του υπάρχουν έτοιμες συναρτήσεις, τις οποίες θα χρησιμοποιήσουμε. Όλες αυτές υπάρχουν μέσα στο αρχείο Servo.h το οποίο θα πρέπει να συμπεριλάβουμε στην αρχή του προγράμματος.

#include <Servo.h>

**Servo myservo**; //σαν τις μεταβλητές

void setup (){

myservo.attach(**X**); //position of the servo on D9

}

void loop(){

}

**Αποστολή**

Συνδέστε στο Arduino έναν σερβοκινητήρα και κάντε τον να:

1. Κινείται μια φορά μόνο σε κάποια επιθυμητή γωνία (0,90,180 πχ) που θα αλλάζει μέσω του προγράμματος κάθε φορά
2. Κινείται διαρκώς από μια γωνία (πχ 0 μέχρι 180) και στη συνέχεια να επιστρέφει στην αρχική του θέση
3. Κινείται μια φορά μόνο σε κάποια επιθυμητή γωνία (0,90,180 πχ) που θα αλλάζει μέσω της σειριακής οθόνης
4. Το serial input του σερβο να:
   1. Μην δέχεται μοίρες <0 και >180
   2. να τυπώνει την τωρινή θέση

**Joystick**

**Chart

Description automatically generated**Το joystick μπορείτε να το σκεφτείτε σαν δύο ενωμενα ποτενσιόμετρα. Με το ένα μετράμε την κίνηση στον οριζόντιο άξονα X και με το άλλο την κίνηση στον κάθετο άξονα Υ. Επίσης μπορεί να λειτουργήσει και σαν κουμπί πατώντας το προς τον άξονα Ζ (Ψηφιακή είσοδος).

A picture containing shape

Description automatically generated

Όπως έχουμε αναφέρει παλιότερα το Arduino διαθέτει ανάλυση 10bits του αναλογικού σήματος, άρα μπορεί να μας δώσεις τιμές σε κάθε αναλογική έξοδο. Συνδέοντας το joystick όπως στην παρακάτω εικόνα μπορούμε να δούμε τις διπλανές μετρήσεις.

A picture containing electronics, circuit, electronic engineering, electronic component

Description automatically generatedΟι θύρες του είναι οι:

**G**: Ground

**V**: Τάση

**B**: Button (Digital)

**X**: μετρηση X (analog)

**Y**: μέτρηση Υ(analog)

**Συνδεσμολογία**

A picture containing electronics, circuit, electronic engineering, electronic component

Description automatically generated

|  |  |
| --- | --- |
| Right X | A2 |
| Right Y | A5 |
| Right Z | 7 |
| Left X | A3 |
| Left Y | A4 |
| Left Z | 6 |

**Αποστολή**

Συνδέστε ένα από τα δύο joystick αναλόγως, και προγραμματίστε το Arduino να:

1. Δείχνει στη σειριακή οθόνη τις τρεις τιμές του joystick (x,y,button) **διαρκώς**
2. Το ίδιο με την 1. αλλά **μόνο** αν κάποια από τις 3 τιμές αλλάξει.

**Αποστολή**

Στη συνέχεια προγραμματίστε το Arduino ώστε να δείχνει διαρκώς τις παρακάτω τιμές στη σειριακή οθόνη με ακριβώς την ίδια δομή:

A screenshot of a computer

Description automatically generated with medium confidence

**ΕΔΩ ΛΙΓΗ ΘΕΩΡΙΑ ΓΙΑ FUNCTIONS**

**Read των joystick με συνάρτηση(Επίπεδο 1)**

1.Η συνάρτηση θα παίρνει σαν είσοδο τις θύρες που θέλουμε να διαβάσουμε κάθε φορά, δηλαδή:

**Joy\_read(right\_X,right\_Y,right\_key)**

Στη συνέχεια θα πρέπει να διαβάζει και **να αποθηκεύει** τις τιμές των αναλογικών ή ψηφιακών εισόδων, άρα θα πρέπει να:

………………………………………………………………………………………………………………………………………………….

Τέλος θέλουμε να τυπώνει στη σειριακή οθόνη το αντίστοιχο μήνυμα.

**Read των joystick με συνάρτηση(Επίπεδο 2)**

Για να μπορέσουμε να τυπώσουμε τις τιμές **και** των δύο joystick (**left/right)** θα πρέπει η συνάρτηση να δέχεται μια **νέα** μεταβλητή με την οποία θα αλλάζουμε το περιεχόμενο της σειριακής οθόνης. Δηλαδή:

|  |
| --- |
| Serial.print(test);  Serial.print("\_X = " |

Η μεταβλητή test θα αλλάζει κάθε φορά ανάλογα τι θέλουμε να τυπώσουμε.

**Αν test=”right” τότε θα τυπωθεί right\_X ενώ αν test=”left” θα τυπωθεί left\_X**

Για να το κάνουμε αυτό θα πρέπει η συνάρτηση να δηλωθεί ως εξής:

|  |
| --- |
| void joy\_read(……. , ……. , ……., **char test[20])** |

**Τύποι δεδομένων στην C – Arrays**

Ένας πίνακας ή array είναι πολλές μεταβλητές μαζί, τις οποίες μπορούμε να καλέσουμε μέσω ενός αριθμού που λέγεται index. Παράδειγμα:

int myInts[6];

int myPins[] = {2, 4, 8, 3, 6};

int mySensVals[5] = {2, 4, -8, 3, 2};

char message[6] = "hello";

Μπορεί να είναι μεταβλητές τύπου int ή char ή ό,τι άλλο χρειάζεται αλλά πάντα κοινού τύπου. Δεν μπορούμε δηλαδή σε έναν πίνακα αριθμών να βάλουμε γράμματα. Η αρίθμηση ξεκινάει από το μηδέν, δηλαδή το πρώτο στοιχείο κάθε πίνακα βρίσκεται στη θέση 0. Παράδειγμα:

myPins[0] 2

myPins[1] 4

myPins[2] 8

myPins[3] 3

myPins[4] 6

Για να αλλάξουμε την τιμή ενός στοιχείου ενός πίνακα χρησιμοποιούμε το μόνο «ίσον»:

myPins[0] 12

Μπορούμε να χρησιμοποιήσουμε επανάληψη αν θέλουμε να διαβάσουμε όλα τα στοιχεία ενός πίνακα:

for (byte i = 0; i < 5; i = i + 1) {

Serial.println(myPins[i]);

}

Βρείτε και αναλύστε το παράδειγμα που βρίσκεται στο: **Examples/Control/Arrays**

**Read των joystick με συνάρτηση(Επίπεδο 3)**

Τροποποιείστε το πρόγραμμα ώστε να δέχεται μια 4η μεταβλητη. Η μεταβλητή αυτή θα καθορίζει πότε και αν πρέπει να αλλάζει σειρά το τελευταίο Serial.print()

**Λύση αποστολής με joystick read via function**

const int right\_X = A1; // define the right X pin to A2

const int right\_Y = A2; // define the right Y pin to A5

const int right\_key = A0; //define the right key pin to 7（that is the value Z）

const int left\_X = A4; //define the left X pin to A3

const int left\_Y = A5; // define the left Y pin to A4

const int left\_key = A3; //define the left key pin to 8（that is the value Z）

void setup()

{

Serial.begin(9600); // set the baud rate to 9600

}

void loop()

{

joy\_read(right\_X,right\_Y,right\_key,"right",1);

joy\_read(left\_X,left\_Y,left\_key,"left",0);

}

void joy\_read(int w1,int w2,int w3,char test[20],int end){

int x1;

int y1;

int z1;

x1 = analogRead(w1); // read the value of right X

y1 = analogRead(w2); // read the value of right Y

z1 = analogRead(w3); //// read the value of right Z

//Serial.println("\*\*\*\*\*\*\*\*\*\*right\*\*\*\*\*\*\*\*\*\*");

Serial.print(test);

Serial.print("\_X = "); // on the serial monitor, print out right\_X =

Serial.print(x1 ,DEC); // print out the value of right X and line wrap

Serial.print(" ");

Serial.print(test);

Serial.print("\_Y = ");

Serial.print(y1 ,DEC);

Serial.print(" ");

Serial.print(test);

Serial.print("\_key = ");

Serial.print(z1 ,DEC);

if(end==1)

Serial.print(" ||| ");

else

Serial.println("");

}

**The millis() function**

<https://www.norwegiancreations.com/2017/09/arduino-tutorial-using-millis-instead-of-delay/>

Η εντολή delay που χρησιμοποιούμε ως τώρα στα προγράμματά μας, σταματάει το πρόγραμμα για τον αριθμό που θα δηλώσουμε, πχ delay(5000) δηλαδή 5s. Κατά τη διάρκεια αυτού του χρόνου το πρόγραμμα δεν θα ανταποκρίνεται, δεν θα κάνει τίποτα άλλο ταυτόχρονα δηλαδή μέχρις ότου περάσει ο χρόνος που ορίσαμε. Σε περίπτωση λοιπόν που θέλουμε να γίνονται και άλλα πράγματα ταυτόχρονα, δεν μπορούμε να το χρησιμοποιήσουμε.

Για να λύσουμε αυτό το πρόβλημα μπορούμε να χρησιμοποιήσουμε την εντολή **millis()** η οποία μας **επιστρέφει** τον αριθμό των milliseconds που πέρασαν από τη στιγμή που ξεκίνησε το πρόγραμμα.

Παράδειγμα χρήσης millis() αντί για delay()

int period = 500;

unsigned long time\_now = 0;

void setup() {

Serial.begin(9600);

}

void loop() {

if(millis() >= time\_now + period){

time\_now = time\_now + period;

Serial.print("Hello, the time is : ");

Serial.println(time\_now);

}

//Run other code

}

A picture containing text, screenshot, font, document

Description automatically generated**Αποστολή**

Φτιάξε ένα πρόγραμμα που θα τυπώνει 4 διαφορετικά μηνύματα:

1. Το πρώτο μήνυμα κάθε 5s
2. Το δεύτερο μήνυμα κάθε 7s
3. Το τρίτο μήνυμα κάθε 11s
4. Το τέταρτο μήνυμα κάθε 13s

# **Λύση**

#define INTERVAL\_MESSAGE1 5000

#define INTERVAL\_MESSAGE2 7000

#define INTERVAL\_MESSAGE3 11000

#define INTERVAL\_MESSAGE4 13000

unsigned long time\_1 = 0;

unsigned long time\_2 = 0;

unsigned long time\_3 = 0;

unsigned long time\_4 = 0;

void print\_time(unsigned long time\_millis);

void setup() {

Serial.begin(115200);

}

void loop() {

if(millis() >= time\_1 + INTERVAL\_MESSAGE1){

time\_1 +=INTERVAL\_MESSAGE1;

print\_time(time\_1);

Serial.println("I'm message number one!");

}

if(millis() >= time\_2 + INTERVAL\_MESSAGE2){

time\_2 +=INTERVAL\_MESSAGE2;

print\_time(time\_2);

Serial.println("Hello, I'm the second message.");

}

if(millis() >= time\_3 + INTERVAL\_MESSAGE3){

time\_3 +=INTERVAL\_MESSAGE3;

print\_time(time\_3);

Serial.println("My name is Message the third.");

}

if(millis() >= time\_4 + INTERVAL\_MESSAGE4){

time\_4 += INTERVAL\_MESSAGE4;

print\_time(time\_4);

Serial.println("Message four is in the house!");

}

}

//if we want to also print time

void print\_time(unsigned long time\_millis){

Serial.print("Time: ");

Serial.print(time\_millis/1000);

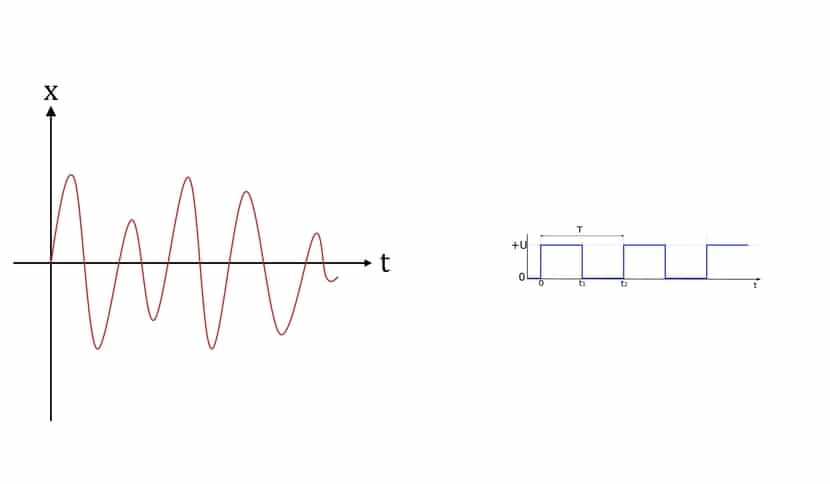
Serial.print("s - ");

Στο επόμενο μάθημα να δείξουμε αναλυτικά:

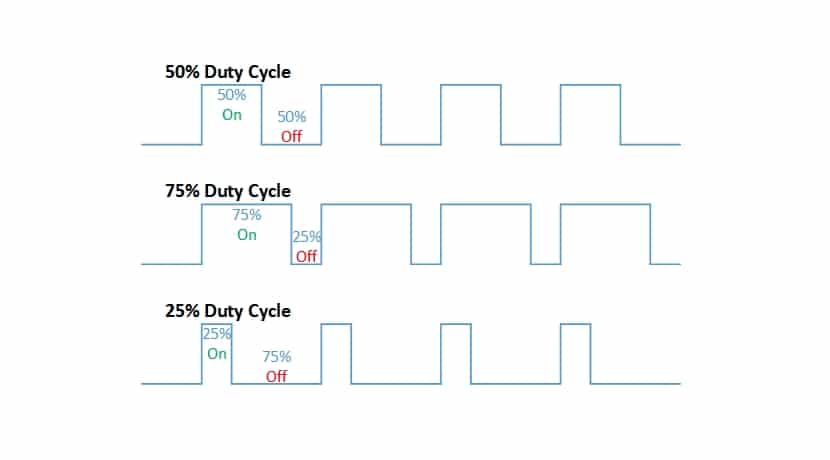
**NΑ ΦΕΡΩ Μ3 >12ΜΜ η παξιμαδια M2.5 M2**

## PWM – Pulse Width Modulation

Οι θύρες PWM έχουμε πει ότι χρησιμοποιούνται ώστε να μιμηθούμε ένα αναλογικό σήμα μέσω ενός ψηφιακού PIN. Τα είχαμε χρησιμοποιήσει όταν ελέγχαμε τη φωτεινότητα ενός LED αλλά με αντίστοιχο τρόπο μπορούμε να τα χρησιμοποιήσουμε για να ελέγξουμε την γωνία ενός σερβοκινητήρα.



Η βασική διαφορά ενός ψηφιακού (δεξιά) με ένα αναλογικό (αριστερά) σήμα είναι ότι το αναλογικό σήμα μπορεί να πάρει οποιαδήποτε τιμή, μέσα σε ένα εύρος τιμών (0-1024) ενώ το ψηφιακό μπορεί να έχει ορισμένες τιμές (0 ή 1). Σαν ένα παράδειγμα, ένα αναλογικό σήμα θα μπορούσε να είναι η στάθμη του νερού σε ένα δοχείο κάθε στιγμή ενώ ένα ψηφιακό θα ήταν η πληροφορία έχει/δεν έχει νερό στο δοχείο.



Με το PWM (Pulse Width Modulation) χρησιμοποιώντας ένα ψηφιακό σήμα (ON/OFF) και ρυθμίζοντας την διάρκεια του καθενός να προσομοιάσουμε ένα αναλογικό σήμα. Δηλαδή ορίζουμε για πόση ώρα θα είναι HIGH και πόση ώρα θα είναι LOW το σήμα και έτσι μπορούμε να αλλάξουμε την φωτεινότητα του LED, ή αντίστοιχα την γωνία του σερβοκινητήρα.

**Παράδειγμα αλλαγής φωτεινότητας LED αλλά με digitalWrite**

int ledPin = 9;

int brightness = 0;

int fadeAmount = 5;

void setup() {

pinMode(ledPin, OUTPUT);

}

void loop() {

digitalWrite(ledPin, HIGH); // turn LED on

delayMicroseconds(brightness); // delay for "on" phase of cycle

digitalWrite(ledPin, LOW); // turn LED off

delayMicroseconds(255 - brightness); // delay for "off" phase of cycle

brightness = brightness + fadeAmount;

if (brightness <= 0 || brightness >= 255) {

fadeAmount = -fadeAmount;

}

}

Το πρόγραμμα αυτό:

digitalWrite(ledPin, HIGH); // turn LED on

delayMicroseconds(brightness); // delay for "on" phase of cycle

digitalWrite(ledPin, LOW); // turn LED off

delayMicroseconds(255 - brightness); // delay for "off" phase of cycle

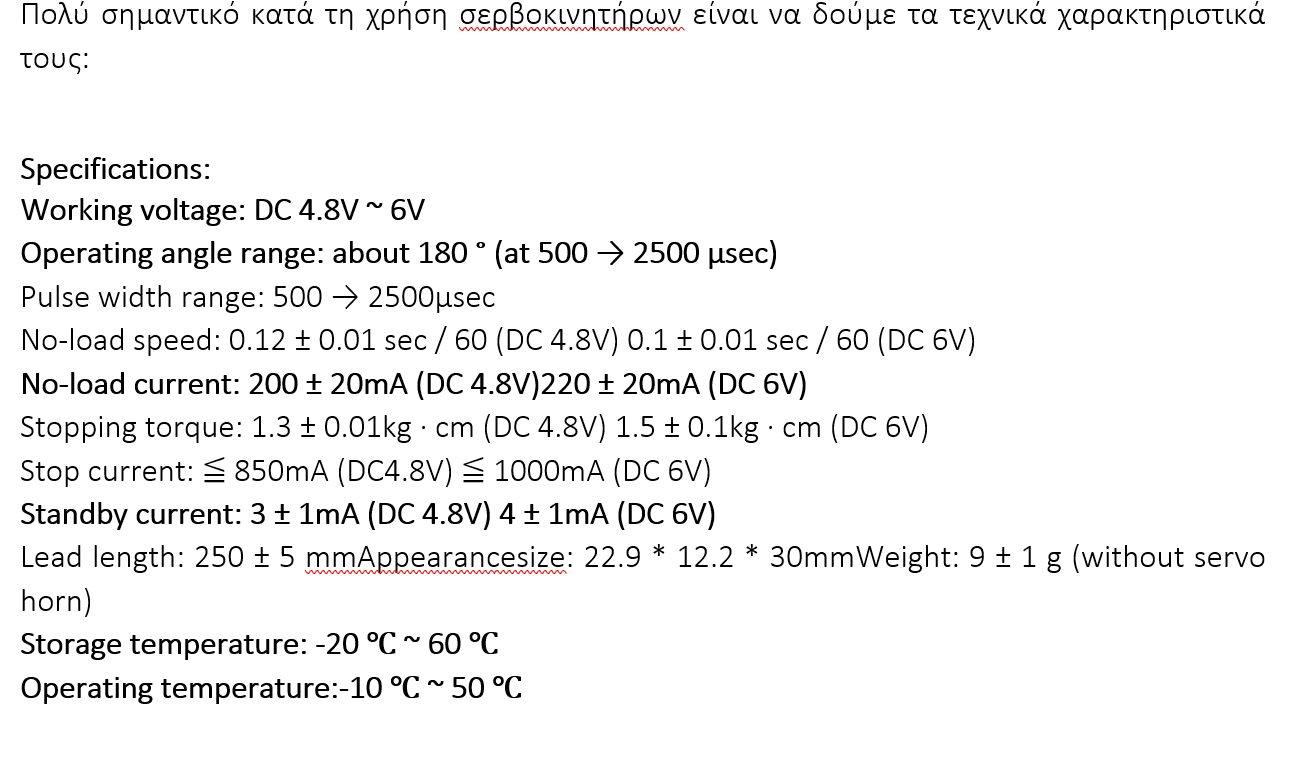
έχει ακριβώς την ίδια λειτουργικότητα με το να γράφαμε analogWrite(pin,value).

Αντίστοιχα μπορούμε για να ορίσουμε την γωνία ενός σερβοκινητήρα να κάνουμε το εξής:

|  |
| --- |
| int servopin=A0;//Define digital interface A0 to connect servo steering gear signal line  int myangle; //Define angle variable  int pulsewidth; //Define pulse width variable  void setup()  {  pinMode(servopin,OUTPUT); //Set steering gear interface as Output  }  void servopulse(int servopin,int myangle)  {  pulseWidth = map(angle, 0, 180, 500, 2500);  // or  //pulsewidth=(myangle\*11)+500; //Converts the Angle to a pulse width value //of 500 – 2480  digitalWrite(servopin,HIGH); //The steering gear interface level is high  delayMicroseconds(pulsewidth); //The microsecond number of the delay pulse width value  digitalWrite(servopin,LOW); //Turn the steering gear interface level to low  delayMicroseconds(20000 - pulseWidth);  }  void loop()  {  servopulse(servopin,90); //Set steering gear Angle  } |

**int pulseWidth = map(angle, 0, 180, 500, 2500);**

Αυτή η εντολή μετατρέπει την μεταβλητή **angle** η οποία αρχικά έχει τιμή από 0 έως 180 σε τιμή από 500 έως 2500. Οι αριθμοί αυτοί 500 – 2500 δεν είναι τυχαίοι, αλλά αναφέρονται στο datesheet κάθε σερβοκινητήρα. Αν βάλουμε διαφορετικά όρια από αυτά που ορίζει η εταιρία κατασκευής υπάρχει περίπτωση να καταστραφεί ο κινητήρας.





**Αποστολή**

delayMicroseconds(20000 - pulseWidth);

Αντικαταστήστε την παραπάνω εντολή με την εντολή delay().

# Αποστολή (Προϋποθέτει ολοκληρωμένη κατασκευή)

Το ρομπότ μετά το ανέβασμα του κώδικα θα να κάνει κυκλικά (επανάληψη) την ακόλουθη αλληλουχία κινήσεων:

1. Στροφή βάσης δεξια
2. Η δαγκάνα ανοίγει
3. Επέκταση
4. Κατέβασμα
5. Δαγκάνα κλεινει
6. Επιστροφή
7. Ανέβασμα
8. Στροφή αριστερα
9. Επέκταση
10. Κατέβασμα
11. Δαγκάνα κλεινει
12. Επιστροφή
13. Ανέβασμα

**Έλεγχος του βραχίονα μέσω των joystick**

Αποστολή

Προγραμματίστε το Arduino ώστε:

1. Να διαβάζει όλες τις μετρήσεις των δύο joystick (left, right)

Ονομάστε τις μεταβλητές με βάση τον παρακάτω πίνακα.

A picture containing text, screenshot, font, number

Description automatically generated

1. Προγραμματίστε τους σερβοκινητήρες με βάση των παραπάνω πίνακα.

Στον κώδικα του Project 5 θα βρείτε το πρόγραμμα ελέγχου του βραχίονα μέσω joystick. Ας δούμε τι κάνει περιληπτικά το πρόγραμμα αυτό:

1. **loop()**

Διαβάζουμε τις (3+3) τιμές των δύο joystick:

|  |
| --- |
| x1 = analogRead(right\_X); // read the value of right X axis  y1 = analogRead(right\_Y); // read the value of right Y axis  z1 = digitalRead(right\_key); ////read the value of right Z axis    x2 = analogRead(left\_X); //read the value of left X axis  y2 = analogRead(left\_Y); //read the value of left Y axis  z2 = digitalRead(left\_key); //read the value of left Z axis |

Και στη συνέχεια καλούνται 4 συναρτήσεις:

|  |
| --- |
| **grip()**  **rotate\_motion();**  **right\_ser();**  **left\_ser();** |

Μπορείτε να περιγράψετε περιληπτικά τι κάνει κάθε συνάρτηση από τις παραπάνω;