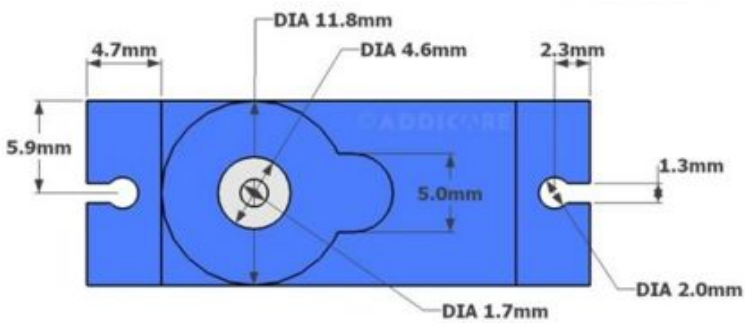
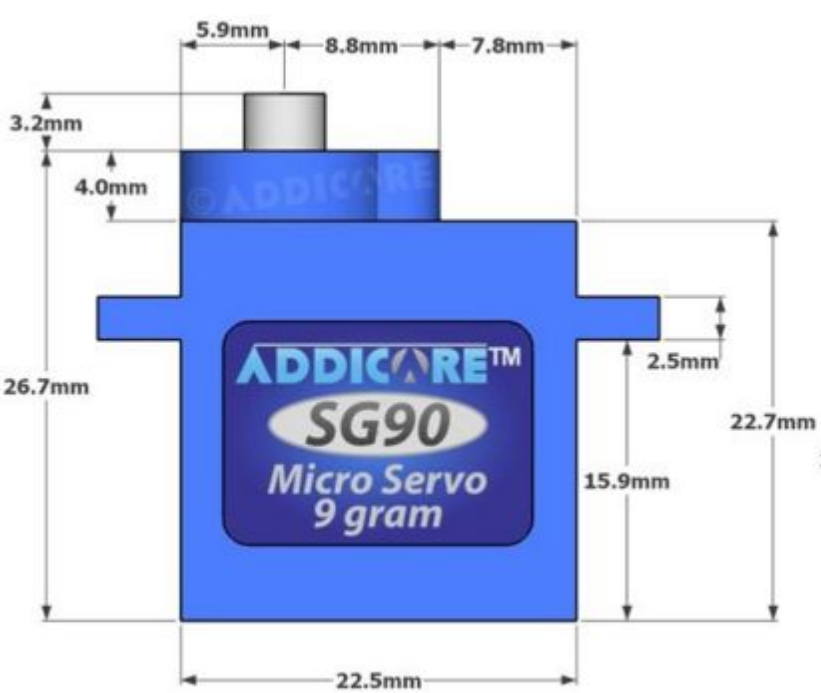
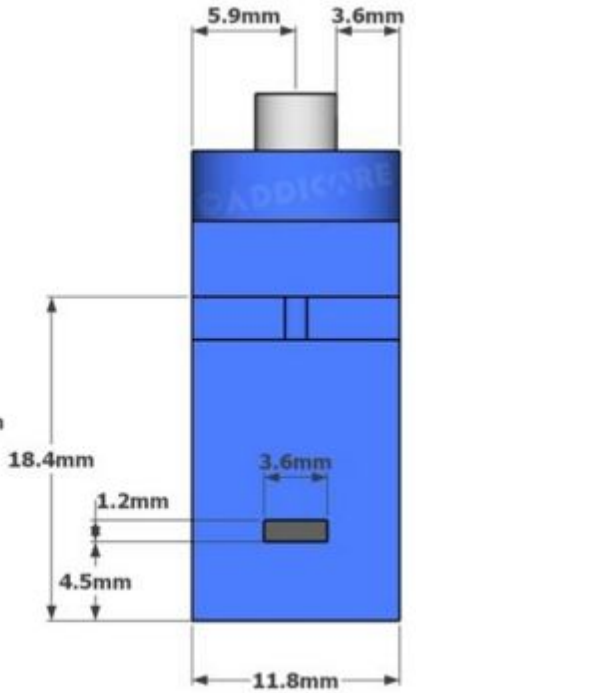
Servo motor:

A szervo motorokat PWM jelekkel lehet vezérelni. Mi most az SG90-es motort fogjuk használni, amelynek 3 lába van. A piros vezetékre az 5V-ot, a barna vezetékre a 0V-os földet, a narancssárgára pedig a vezérlő jelet kell kötni. Az Arduiono-nak van beépített Servo library könyvtára, amit felhasználhatunk a vezérléshez. Mi most egy potenciométert fogunk használni a vezérléshez. Ez a szervo 180°-ban tud mozogni, 90°-ot balra, 90°-ot jobbra. Megoldható, hogy 360°-ban körbe forogjon, de ahhoz szét kell szerelni és egy feszültségosztót kell beleforrasztani ellenállások segítségével, illetve az egyik fogaskerékről le kell törni a fizikai védelmet biztosító pöcköt. Ez egy motorvezérlő építése nélkül használható kis teljesítményű servo.



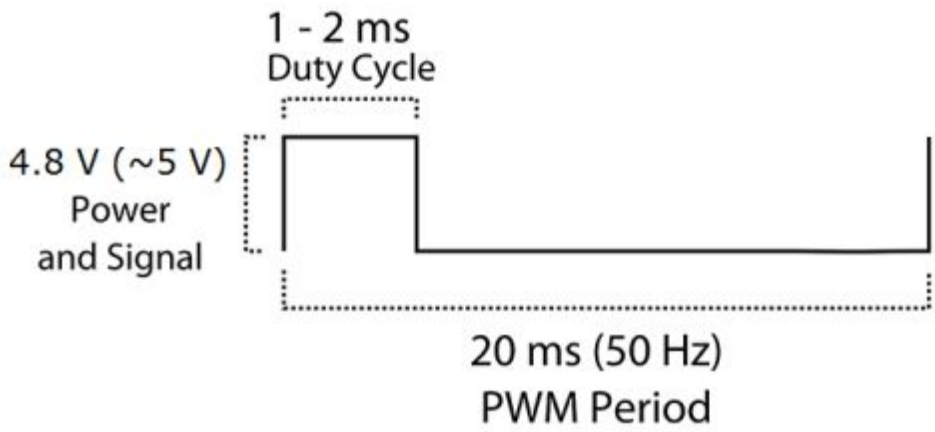




- Weight: 9 g
- Dimension: 22.2 x 11.8 x 31 mm approx.
- Stall torque: 1.8 kgf·cm
- Operating speed: 0.1 s/60 degree
- Operating voltage: 4.8 V (~5V)
- Dead band width: 10 µs
- Temperature range: 0 ºC – 55 ºC

A szögsebessége 0,1s/60°, ami azt jelenti, hogy 0,1 másodperc kell ahhoz, hogy 60°-ot forduljon. Ebből több mindent kiszámíthatunk.
Például ahhoz, hogy 360° forduljon, 360°/60°=6-szor 60°-ot kell fordulnia, ami 6-szor annyi időbe telik, azaz 6*0,1=0,6 másodpercig tart.
Vagy azt, hogy 1 perc alatt, ami 60sec, 60s/0,1s=600-szor 60°-ot tud fordulni, ami 36000° percenként.
Azaz 36000°/60s=600°-ot másodpercenként. Ez egyben a szögsebessége is w=500°/s=500 1/s.
Ha 36000°-ot tesz percenként, akkor a 360°-ot, azaz egy teljes fordulatot, 36000°/360°=100 fordulatot tesz meg percenként, ami azt jelenti, hogy 100rpm-es.

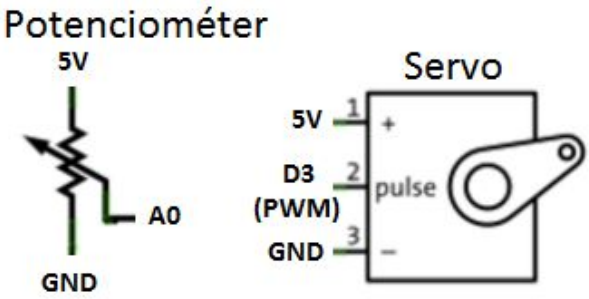
A forgási nyomatéka 1,8kgf*cm.
Tudjuk, hogy az erő mértékegysége a Newton, azaz kg*m/s^2, azaz a tömeg és a gyorsulás szorzata. Ha tudjuk egy testről, hogy 1kg tömegű, és a gravitációs gyorsulás 9,8m/s^2 , azaz körülbelül 10m/s^2, akkor a gravitációs mező 1kg*10m/s^2=10N munkát végez. Tehát ha a kg-ot durván N-ra szeretnénk átváltani, akkor 10 a váltószám. Cm-ről méterre pedig 100-al kell osztani. Így, 1,8*10/100=0,18Nm. Ha valamihez hasonlítanunk kéne a motor forgatásának erejét, akkor 1Nm forgatóerőt úgy tudunk elképzelni, hogy egy 10dkg tömegű testet - amire körülbelül 10m/s^2 gravitációs gyorsulás hat, és amit mi a kezünkben tartunk 1m-re a testünktől - a gravitáció ekkora erővel próbál kicsavarni a kezünkből. Az 1Nm-ben a 0,18Nm 1/0,18=5,56-szor van meg, tehát ha a 10dkg-os testet kicseréljük egy 10/5,56=1,79~1,8dkg-osra, akkor a fenti példánk ugyanúgy helyt áll. De ha megtartjuk a 10dkg-ot, és a távolságot változtatjuk 1m helyett 1m/5,56=0,1798~18cm-re, akkor a példánk ugyanúgy jó. Látszik, hogy a példában minden változhat, ez csak egy viszonyítási szám, ami az elforgatni kívánt tömegtől, a forgatási ponttól való távolságtól, és a gravitációs gyorsulástól is függ.

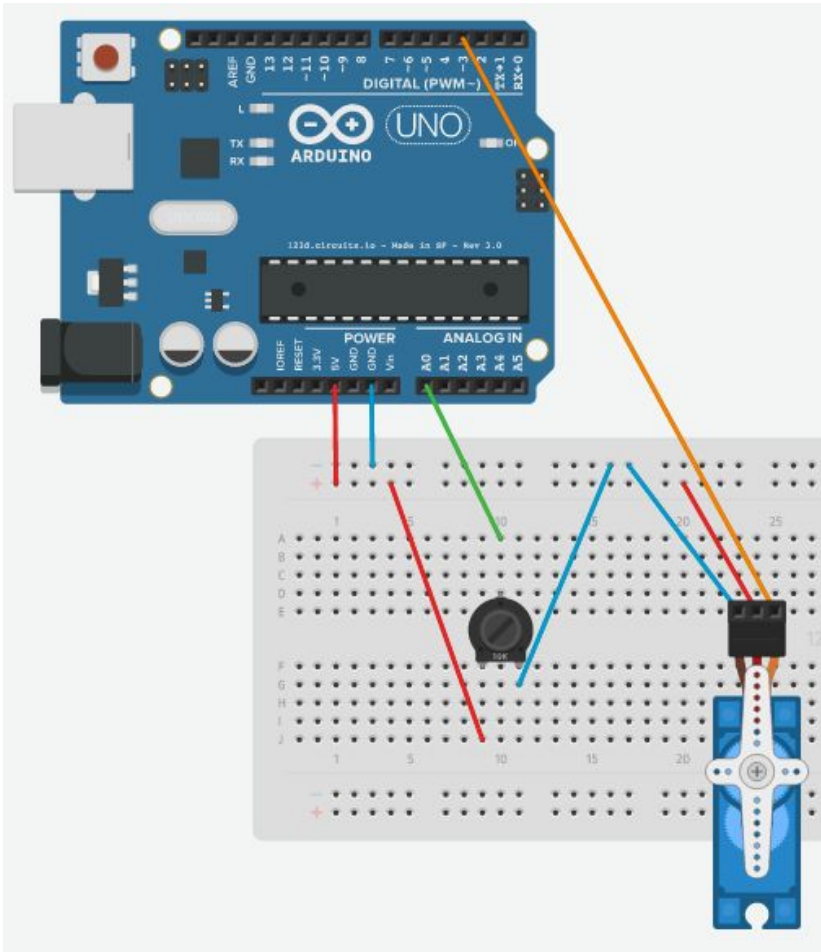


A PWM jel periódusa 20ms-os, ami a motort vezérli. 1/20ms=1/0,020s=50Hz-es frekvenciának felel meg. Ebből a PWM periódusból a kitöltöttség nagyon kicsi 1-2ms, azaz magas ideje/összes idő*100=1/20*100=5% 1 ms-esetén, akkor 2ms-esetén 10%, tehát összeségében azt mondhatjuk, hogy 5-10%-os kitöltöttségű 50Hz-es PWM jel.

A kapcsolás:

Egy potenciométert kötünk be az 5V és a föld közé. Ahogy tekergetjük, az ellenállása ennek megfelelően változik, hiszen egy változtatható ellenállást testesít meg. A harmadik lábát egy analóg portra kell kössük az Arduino-n. Az itt leolvasott feszültségszintet egy 0 és 1023 közötti skálán fogjuk megkapni. És ennek segítségével úgy fogjuk állítani a vezérlőjelet, hogy ezt az analóg értéket átvetítjük egy 0 és 179 közé eső skálára, ami majd azt a szöget határozza meg nekünk, hogy milyen szögbe álljon a servo motorunk. A servo motort is bekötjük az 5V és a föld közé. És a harmadik lábát egy olyan digitális portra kötjük, ami PWM képes, ezzel fogjuk szabályozni a motort, hogy milyen irányba álljon.





A kód:

A kód elején behívjuk az Arduino szervó vezérlő library-jét. Példányosítunk egy **Servo** objektumot. A setup részben hozzárendeljük a vezérlőjel portjához az **attach** metódussal, erre a lábra küldjük majd ki a PWM jelet. Az attach-nak van egy 3 paraméteres változata is, ahol a második egy minimum, a harmadik pedig egy maximum értéke a us-oknak, amit majd megadhatunk a write metódusnak. A loop részben pedig a **write** metódus segítségével megadjuk neki, hogy merre milyen irányba kell átálnia. A write metódusnak, ha 200 alatti értéket adunk meg, akkor °-nak veszi, különben pedig a PWM jel pulzus szélességének, azaz, hogy hány us-ig, magas a jel. Ha alapjáraton us-ben akarjuk megadni, akkor használhatjuk a **writeMicroseconds** metódust. A **detach** metódussal lecsatlakoztathatjuk a motort. az **attached** metódussal pedig azt vizsgálhatjuk, hogy fel van-e csatlakoztatva egy portra. A loop ciklus 15ms-ot vár, ez ahhoz kell, hogy a motor elérjen a potenciométer által meghatározott szögbe. Ahogy fent kitárgyaltuk, ez a motor 0,6 másodperc alatt fordulna egy teljes kört, ha tudna és nem lenne lekorlátozva 180°-ra. Tehát körülbelül 0,3 másodperc, azaz 300ms-kell ahhoz, hogy az egyik végállapotából a másikba biztosan elérjen. Ez a 300ms a legroszabb esettel számol, amikor a motornak a legnagyobb utat kell bejárnia, azaz az egyik széléről a másik szélére kell eljutnia. De ha 15ms helyett a loop-ot 300ms-ra állítanánk, akkor hiába tekernénk gyorsan a potenciométert, a program megvárná azt az időt, amíg a motornak biztosan oda kell érnie a jelenlegi pozíciójából a beállítottra és közben nem reagálna arra, hogy a potenciométeren mit állítunk. Még akkor is 300ms-ot várna, ha csak 1°-ot kell a motornak arrébb forognia. A 15ms alatt 600ms/15ms=40-ed részét tudja fordulni a teljes 360°-nak, azaz 360°/40=9°-ot. Ha ennél nagyobbat tekerünk, és még a **15ms-on belül** eltekerjük a potenciométert, akkor az első pozícióra még nem is tud beállni teljesen, csak **9°-ot tud** felé mozdulni, és máris a második beállítás felé forog tovább.

```
#include <Servo.h>

int poti = A0;
int vezerlojel = 3;
Servo motor;

void setup() {
  motor.attach(vezerlojel);
}

void loop() {
  motor.write(map(analogRead(poti),0,1023,0,179));
  delay(15);
}
```

A motorunkról **nem tudjuk lekérdezni, hogy éppen milyen szögben áll** miközben forgatunk. Tehát ahhoz, hogy **biztosak legyünk, hogy beállt egy szögbe**, miután kiadtuk neki a write utasítást, várnunk kell 300ms-ot, a következő utasításig. A következő kód például beállítja a motort 90°-ba, majd 0-ba, majd 180-ba, és ha körbe járt, akkor ismét vissza állítja 180-ból 90-be, majd 0-ba és így tovább.

```
void loop() {
  motor.write(90);
  delay(300);
  motor.write(0);
  delay(300);
  motor.write(180);
  delay(300);
}
```

Ha a célunk viszont az, hogy a motor folyamatosan pásztázzon, azaz az egyik szélétől **1°-onként** a másik széle felé mozogjon, majd visza, akkor elég, ha csak 600ms/360°=1,6667ms ~ 2ms-ot vár 1-1 beállítás között.

```
#include <Servo.h>

int szog = 0;
int vezerlojel = 3;
Servo motor;

void setup() {
  motor.attach(vezerlojel);
}

void loop() {
  for(szog=0;szog<=180;szog++){
    motor.write(szog);
    delay(2);
  }
  for(szog=180;szog>=0;szog--){
    motor.write(szog);
    delay(2);
  }
}
```

Természetesen, ha **közben várnunk kell még valamire, mondjuk egy kibocsájtott ultrahang jel visszaérkezésére**, akkor a várakozási időt ennek megfelelően kell megnövelni. Az általunk használt ultrahangos szenzor több jelet küld ki, majd ennek mediánját adja vissza. Erre azért van szükség, mert egyetlen mérés nagyon pontatlan lehet, sok minden megzavarhatja az ultrahangot. Ha a mediánt adjuk eredményül, azaz ha sorba rendezzük a kapott távolság eredményeket és kivesszük a sorozat középső elemét, az elég jól reprezentálja a távolság tényleges értékét. Az átlag azért nem jó, mert ha a mért eredmények között van egy nagyon kiugró érték, az nagyon elcsalja a végeredményt rossz irányba. A módusz, azaz a leggyakrabban előforduló érték pedig azért nem jó, mert lehet, hogy a sorozatban minden érték eltér a másiktól kicsit, hiszen századnyi különbségeket is megengedünk, így minden érték csak egyszer szerepel és nincs olyan ami gyakoribb lenne, tehát nagyobb esélyjel lenne a tényleges távolság. A medián mérés viszont több jelből áll, és az utolsó jel visszaérkezésére a ping-nél 29ms-ot kell várni. Így tehát, ha minden °-ban szeretnénk egy mérést, akkor minimum 29-30ms-ot kell pluszban várnunk a forgatások között. Ilyenkor az az idő, ami a között telik el, amíg az egyik pontot lemérjük, majd elmegyünk a széléig, visszafordulunk, majd **ismét elérjük a pontot**, **legrosszabb esetben**, azaz ha a kiindulópont a szélén van, akkor (2+29)ms*180°*2=11160ms, azaz **11,16 másodperc**.

```
void loop() {
  for(szog=0;szog<=180;szog++){
    motor.write(szog);
    delay(2); //amíg szögbe áll
    //mérés
    delay(29); //várunk a jel visszaérkezésére
  }
  for(szog=180;szog>=0;szog--){
    motor.write(szog);
    delay(2); //amíg szögbe áll
    //mérés
    delay(29); //várunk a jel visszaérkezésére
```

```
  }
}
```