Deney 4: Algılayıcı/Güncelleyici Uygulamaları

Deneyin Amacı:
Arduino kartı ile deney tablası kullanarak algılayıcı(sensor) ve güncelleyici(actuator) bağlantısını yapmayı, uygulama devresini kurmayı, giriş/çıkış portlarının kullanılmasını öğrenmek için algılayıcı/güncelleyici devre ve modüllerinin çalışma yöntemlerinin, donanımlarının incelenmesi ve programlanmasının yapılmasıdır.

Deney Öncesi Yapılacak İşlemler:
Bu deneyde kullanılan algılayıcı, DC Motor ve Servo motor elektrik-elektronik devre malzemelerinin işlevleri, kullanım özellikleri, malzemenin fiziksel görünümü, adının ve değerinin üzerindeki verilerden ve veri sayfalarından (datasheets) okunarak elde edilmesi öğrenilecektir. Burada verilen uygulama programları incelenecek, her komut satırının işlevi, açıklaması anlaşılacaktır.

Deneyde Yapılacak İşlemler:
Burada verilen algılayıcı ve güncelleyicilerin nasıl bağlanacağı ve bir Arduino ile nasıl sürüleceği, örnek uygulama programları ile gösterilecektir.

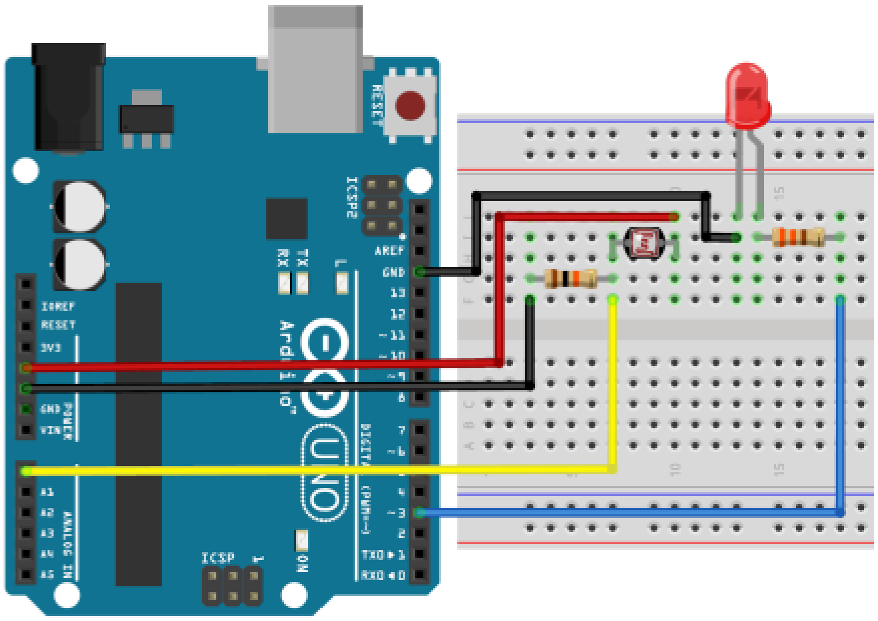
4.a) Algılayıcı (LDR Sensor) Uygulaması (Otomatik Lamba)

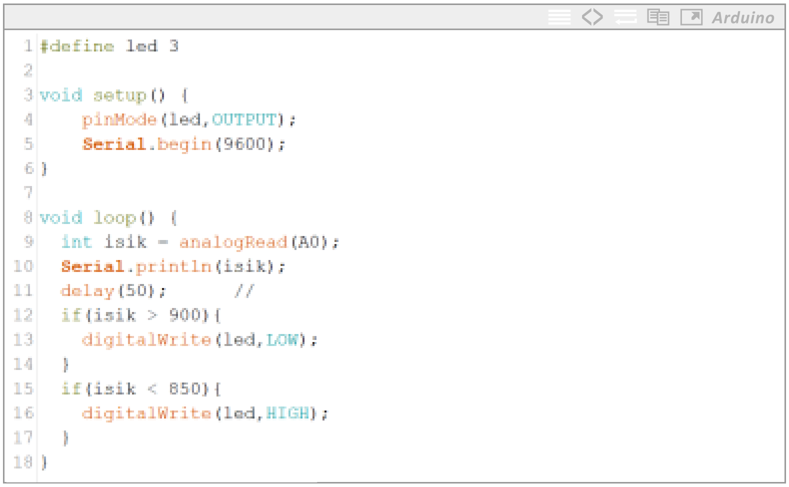
.

Gerekli malzemeler:

- Arduino Uno
- Breadboard
- 5 Adet Erkek-Erkek Jumper Kablo
- 330 Ohm Direnç (Turuncu-Turuncu-Kahverengi)
- 10K Ohm Direnç (Kahverengi-Siyah-Turuncu)
- 5mm Kırmızı LED
- 5mm LDR

Bu uygulamamızda ortamdaki ışığı algılayabilen LDR'den veri okuyup, bu veriye göre LED'imizi yakıp söndüreceğiz. LDR (Light Dependent Resistance) yani fotodirenç ortamdaki ışık miktarına göre direncini değiştirir. Bu direnç değişimini Arduino kartı ile algılayabiliriz.Bu sayede ortamdaki ışık miktarını bilebildiğimiz için ortam karanlık olduğunda LED'i yakıp, aydınlık olduğunda LED'i söndürerek otomatik bir lamba yapacağız. Aynı zamanda aldığımız verileri bilgisayara gönderip, serial monitör üzerinde de görüntüleyeceğiz. Hemen devremizi kurarak başlayalım.





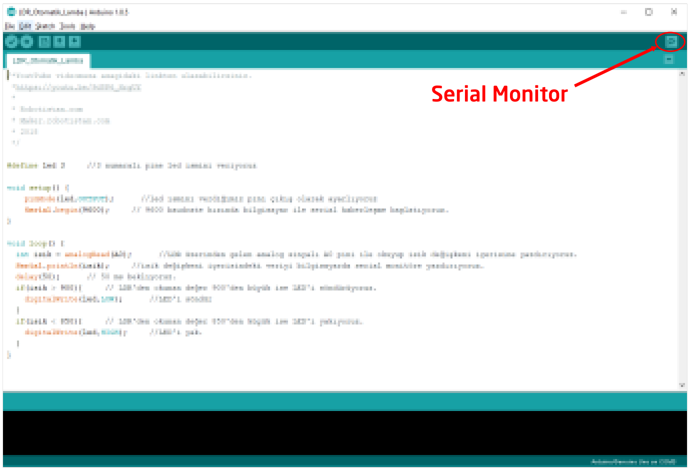
```
#define led 3

void setup() {
    pinMode(led,OUTPUT);
    Serial.begin(9600);
}

void loop() {
  int isik = analogRead(A0);
  Serial.println(isik);
  delay(50);       //
  if(isik > 900){
    digitalWrite(led,LOW);
  }
  if(isik < 850){
    digitalWrite(led,HIGH);
  }
}
```

Kod kısmına geçecek olursak ilk satırımızda LED'i bağlayacağımız pine isim veriyoruz. Bu işlemi "#define" komutu ile yapacağız. Bu işlemden sonra artık ihtiyaç halinde 3 yazmak yerine "led" yazarak işlemleri kolaylaştıracağız.

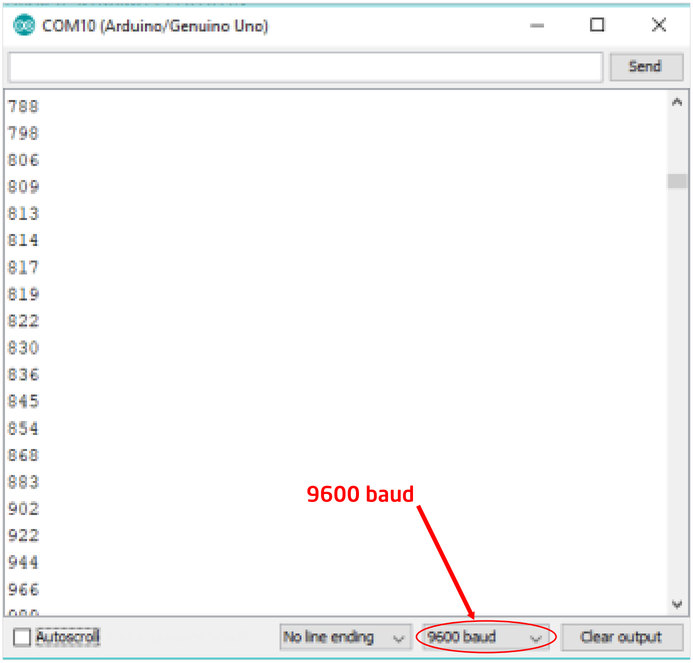
Kodun "setup" kısmında LED'imizi bağladığımız pini çıkış vermemiz ve seri haberleşmeyi başlatmamız gerekiyor. Seri haberleşmeyi 9600 baudrate hızında başlatıyoruz. Bu sayı bilgisayar ve Arduino kartının ne kadar hızlı haberleştiğini belirler. Bu sayıyı rasgele yazamıyoruz. Önceden belirlenmiş hızları kullanmamız gerekmektedir. 300,600,1200,2400,4800,9600,14400,19200,28800,38400 veya 115200 baudrate hızlarını kullanabilirsiniz. Arduino koduna yazdığınız baudrate değeri bilgisayarda açacağını seri monitör'ün sağ alt köşesindeki hız ile aynı olmalıdır.

Ana algoritmamızın döneceği "loop" içerisinde "int" tipinde ve "isik" isminde bir değişken tanımlayıp içerisinde LDR okuduğumu değeri yazdırıyoruz. Okuduğumuz bu değeri seri haberleşme üzerinden bilgisayara gönderiyoruz. 50 ms kadar bekledikten sonra "if" komutu ile gelen değerin istediğimiz değerlerin altında veya üstünde olup olmadığına göre değerlendirip karar veriyoruz. Ortamdaki ışık az ise LDR üzerinden gelen değer küçülecektir. Bizim ortamımızda 850 değerinden sonra LED'in yanmasını istiyoruz. Ortam aydınlanmaya başlayınca da değer artacağından dolayı 900 değerinden sonra sönmesini istiyoruz.

Kodu kartımıza attıktan sonra Arduino IDE'si içerisinde "Serial Monitor" butonuna tıklayıp gelen verileri görebiliriz. LDR üzerinde elinizi getirdiğinizde ışık şiddeti değişeceği için okudğunuz değerlerde değişecektir.


LDR_Otomatik_Lamba | Arduino 1.8.5
File Edit Sketch Tools Help
LDR_Otomatik_Lamba
Serial Monitor

```
/*YouTube videomuzu aşağıdaki linkten ulaşabilirsiniz.
 *https://youtu.be/[illegible]
 *
 * Robotistan.com
 * [illegible]
 * 2018
 */

#define led 2      //2 numaralı pine led ismini veriyoruz

void setup() {
    pinMode(led,OUTPUT);      //led ismini verdiğimiz pini çıkış olarak ayarlıyoruz
    Serial.begin(9600);     // 9600 baudrate hızında bilgisayar ile serial haberleşme başlatıyoruz.
}

void loop() {
  int isik = analogRead(A0);      //LDR üzerinden gelen analog sinyali A0 pini ile okuyup isik değişkeni içerisine yazdırıyoruz.
  Serial.println(isik);     //isik değişkeni içerisindeki veriyi bilgisayarda serial monitöre yazdırıyoruz.
  delay(50);      // 50 ms bekliyoruz.
  if(isik > 900){     // LDR'den okunan değer 900'den büyük ise LED'i söndürüyoruz.
    digitalWrite(led,LOW);      //LED'i söndür
  }
  if(isik < 850){     // LDR'den okunan değer 850'den küçük ise LED'i yakıyoruz.
    digitalWrite(led,HIGH);     //LED'i yak.
  }
}
```



```
788
798
806
809
813
814
817
819
822
830
836
845
854
868
883
902
922
944
966
```

4.b) DC Motor Basics with Arduino

Learn how to connect and control DC motors with your Arduino board.
Author https://www.tutorialspoint.com/arduino/arduino_dc_motor.htm
In this chapter, we will interface different types of motors with the Arduino board (UNO) and show you how to connect the motor and drive it from your board.

- DC motor
- Servo motor
- Stepper motor

A DC motor (Direct Current motor) is the most common type of motor. DC motors normally have just two leads, one positive and one negative. If you connect these two leads directly to a battery, the motor will rotate. If you switch the leads, the motor will rotate in the opposite direction.

DC Motor

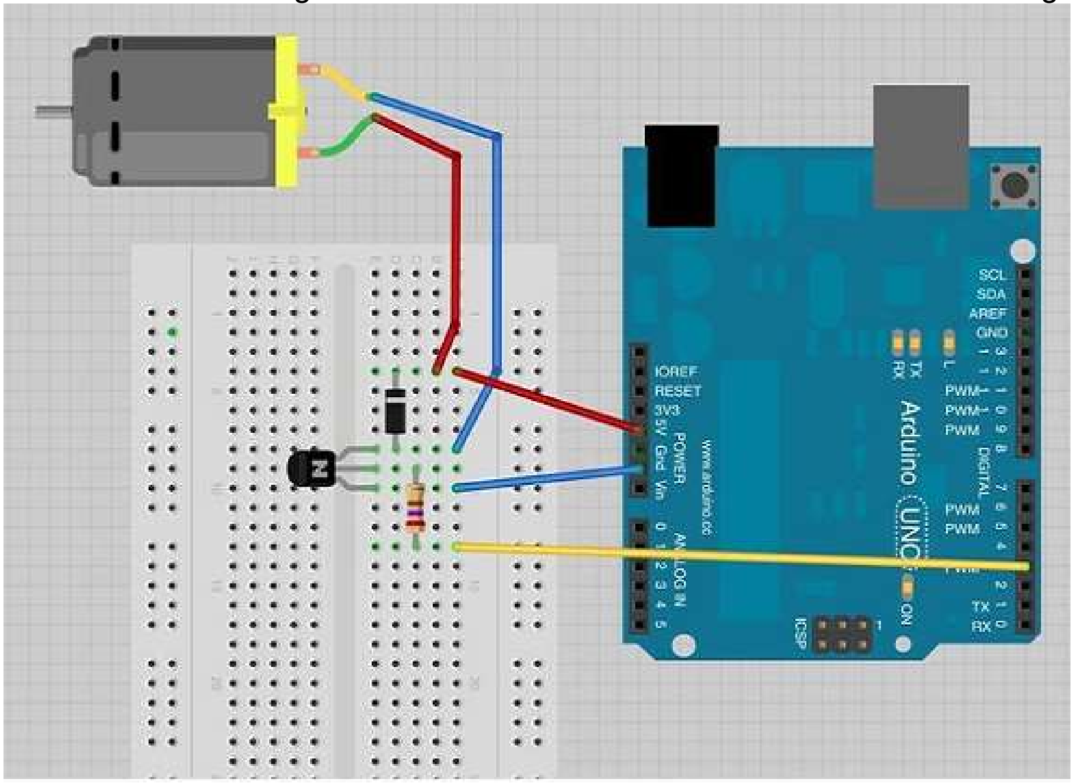
Warning − Do not drive the motor directly from Arduino board pins. This may damage the board. Use a driver Circuit or an IC.
We will divide this chapter into three parts −
Just make your motor spin
Control motor speed
Control the direction of the spin of DC motor

Components Required

You will need the following components −
1x Arduino UNO board
1x BC337 NPN Transistor
1x Small 6V DC Motor
1x 1N4001 diode
1x 270 Ω Resistor

Procedure

Follow the circuit diagram and make the connections as shown in the image given below.



Precautions

Take the following precautions while making the connections.

- First, make sure that the transistor is connected in the right way. The flat side of the transistor should face the Arduino board as shown in the arrangement.
- Second, the striped end of the diode should be towards the +5V power line according to the arrangement shown in the image.

Spin ControlArduino Code

```
int motorPin = 3;

void setup() {
  pinMode(motorPin, OUTPUT);
}

void loop() {
   digitalWrite(motorPin, HIGH);
}
```

Code to Note

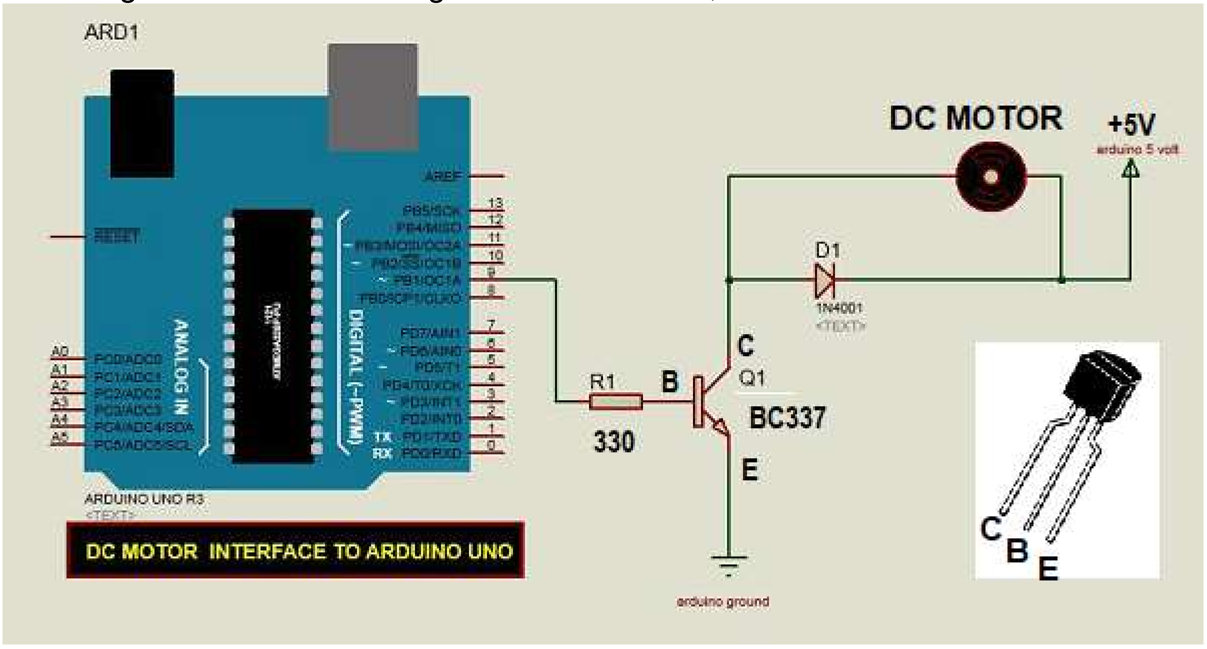
The transistor acts like a switch, controlling the power to the motor. Arduino pin 3 is used to turn the transistor on and off and is given the name 'motorPin' in the sketch.

Result

Motor will spin in full speed when the Arduino pin number 3 goes high.

Motor Speed Control

Following is the schematic diagram of a DC motor, connected to the Arduino board.

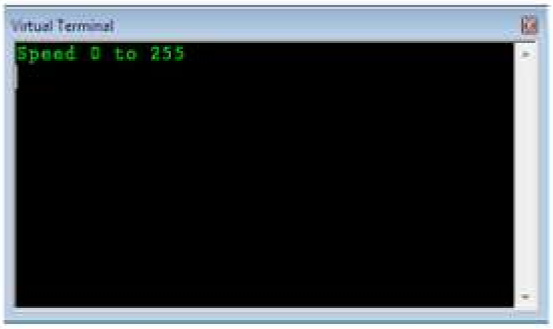


Arduino Code

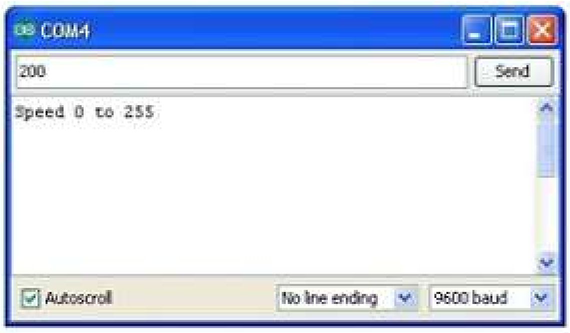
```
int motorPin = 9;
void setup() {
   pinMode(motorPin, OUTPUT);
   Serial.begin(9600);
   while (! Serial);
   Serial.println("Speed 0 to 255");
}

void loop() {
   if (Serial.available()) {
      int speed = Serial.parseInt();
      if (speed >= 0 && speed <= 255) {
         analogWrite(motorPin, speed);
      }
   }
}
```

Code to Note

The transistor acts like a switch, controlling the power of the motor. Arduino pin 3 is used to turn the transistor on and off and is given the name 'motorPin' in the sketch.
When the program starts, it prompts you to give the values to control the speed of the motor. You need to enter a value between 0 and 255 in the Serial Monitor.



```
Speed 0 to 255
```



```
200
Speed 0 to 255
```

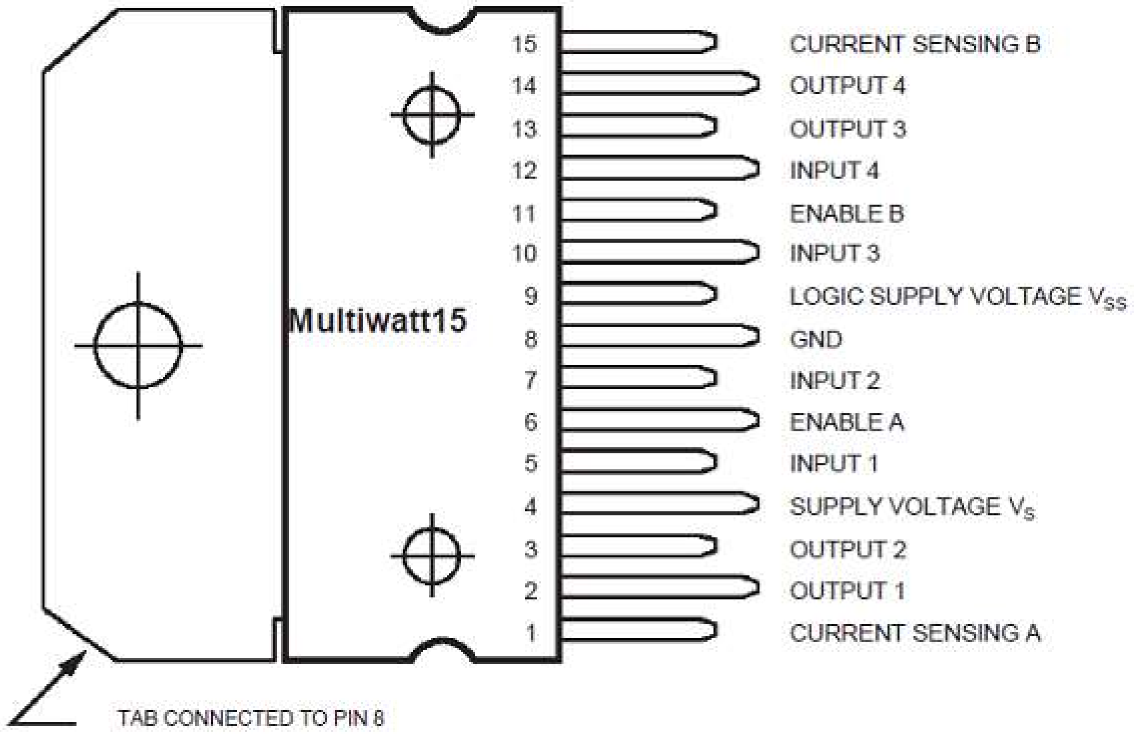
In the 'loop' function, the command 'Serial.parseInt' is used to read the number entered as text in the Serial Monitor and convert it into an 'int'. You can type any number here. The 'if' statement in the next line simply does an analog write with this number, if the number is between 0 and 255.

Result

The DC motor will spin with different speeds according to the value (0 to 250) received via the serial port.

Spin Direction Control

To control the direction of the spin of DC motor, without interchanging the leads, you can use a circuit called an H-Bridge. An H-bridge is an electronic circuit that can drive the motor in both directions. H-bridges are used in many different applications. One of the most common application is to control motors in robots. It is called an H-bridge because it uses four transistors connected in such a way that the schematic diagram looks like an "H."
We will be using the L298 H-Bridge IC here. The L298 can control the speed and direction of DC motors and stepper motors, and can control two motors simultaneously. Its current rating is 2A for each motor. At these currents, however, you will need to use heat sinks.



Components Required

You will need the following components −
1 × L298 bridge IC

Procedure

Following is the schematic diagram of the DC motor interface to Arduino Uno board.
The following table shows which direction the motor will turn based on the digital values of the L298 IC pins IN1 and IN2.

IN1	IN2	Motor Behavior	
0	0	BRAKE	
0	1	FORWARD	CW = Clock Wise
1	0	BACKWARD	CCW = Counter Clock Wise
1	1	BRAKE	

4.c) Servo Motor Basics with Arduino

Learn how to connect and control servo motors with your Arduino board. Author Arduino https://docs.arduino.cc/learn/electronics/servo-motors/ Last revision04.03.2024
The Servo Library is a great library for controlling servo motors. In this article, you will find two easy examples that can be used by any Arduino board.
The first example controls the position of an RC (hobby) servo motor with your Arduino and a potentiometer. The second example sweeps the shaft of an RC servo motor back and forth across 180 degrees.
You can also visit the Servo GitHub repository to learn more about this library.

Hardware Required

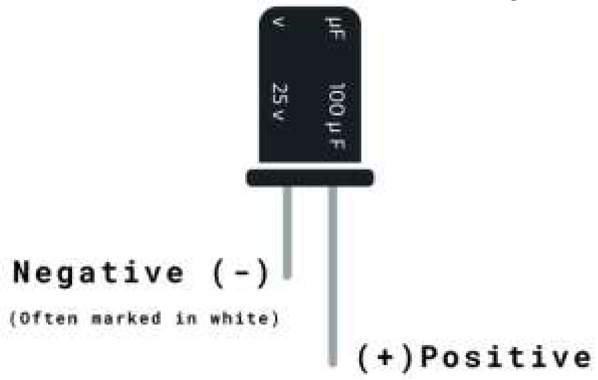
- Arduino Board
- Servo Motor
- 10k ohm potentiometer
- hook-up wires
- capacitors
- power supply

Powering Servo Motors

Servo motors have different power requirements depending on their size and the workload they are experiencing. A common servo motor such as the Feetech Mini Servo Motor requires between 4.8 - 6 V at 5 – 6 mA when idle. It doesn't take very much energy to stand still.
But as soon as the motor starts moving, it starts using more energy, and it gets that energy by pulling more current from the power source.
If it experiences heavier loads such as added weight or an object blocking its movement , it naturally needs to use even more energy to move the obstacle, and as a result the current consumption increases. The current consumption of the motor linked above can reach up to 800 mA.
This high current-draw is generally not safe to draw from an Arduino board. To avoid damaging our board we need to power the servo motor through an external power supply. Choosing the correct power supply depends on the servo motor you are using, so always check the specifications. Pay especially close attention to the:

- **operating voltage range**
- **idle current** - consumption when **not** moving
- **running current** - consumption when moving freely
- **stall current** - consumption under max load or when blocked

To power a 4.8 - 6 V servo you could use a **5 V 1 A** AC Adapter, cut the cable, and connect the wires to the servo using e.g. a breadboard.
Note that USB wall chargers are limited to 500 mA (USB 2.0) or 900 mA (USB 3.0).
If your project needs to move around freely without being attached to a power outlet you can also choose batteries to power the servo. If you need 5 V exactly you can use two 18650 Li-Ion batteries together with a step-down converter.
A step-down converter is needed because 18650 Li-Ion batteries will give you around 7.4 V. The max current depends on the specific battery but most of them are designed to output above 1A which is enough to power our small servo.
If you are using bigger or more servos make sure to check your power requirements accordingly.
Capacitors are recommended for powering servo motors. They help stabilize the power supply, minimize voltage drops, and reduce electrical noise. The specific capacitor values may vary based on the servo motor's requirements, but including them is good practice for better performance and reliability.
When using a Feetech Mini Servo Motor we recommend using a **100 μF** capacitor.



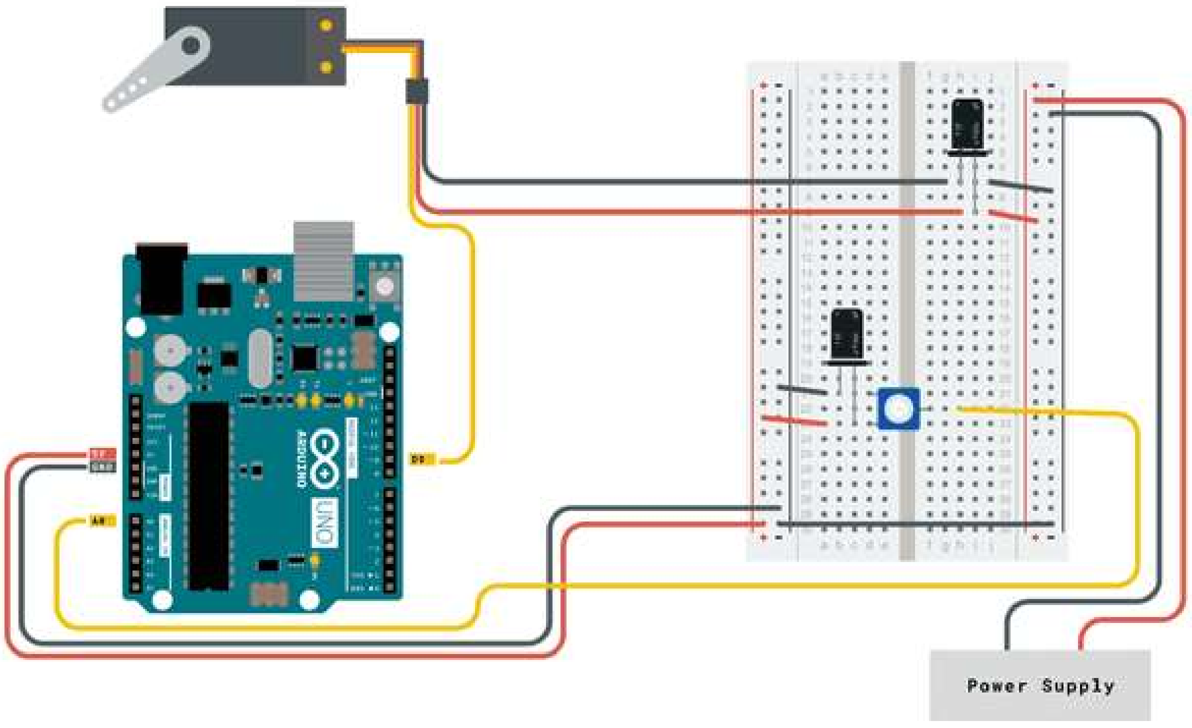
Because some capacitors are polarised (meaning that they have a direction), you may need to be careful with how you connect them to your circuit. Make sure to connect them correctly by checking for markings such as a white stripe, a '+' symbol, or a longer lead. If your capacitor has these, match the indicators of the capacitor with your circuit (pay attention to the + and - signs), and be careful not to exceed the voltage limits. This precaution helps prevent issues like leaks or damage that could harm your circuit.
You can read more about capacitors here.

Circuit

Servo motors have three wires: power, ground, and signal. The power wire is typically red, and should be connected to positive pole (+) of your power source. The ground wire is typically black or brown and should be connected to the negative pole (-) of your power source.
The signal pin is typically yellow or orange and should be connected to PWM pin on the board. In these examples, it is pin number 9.

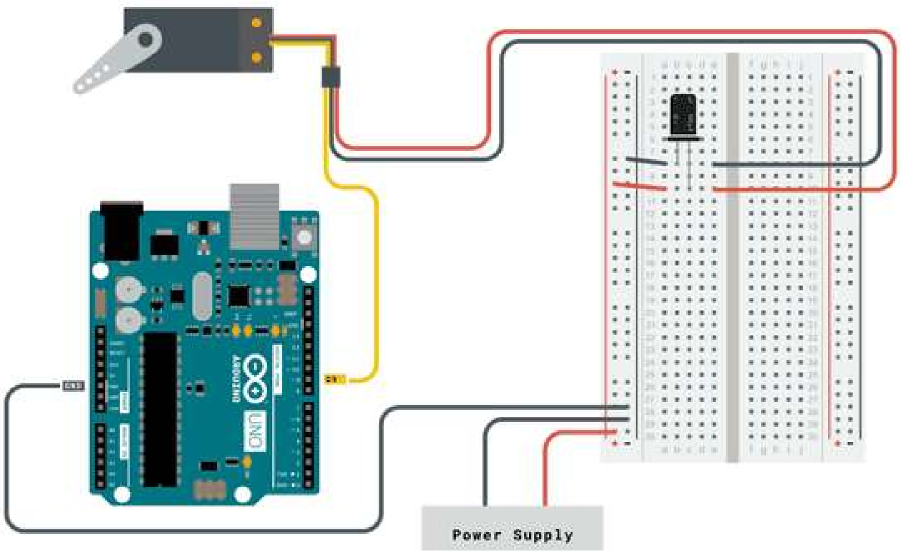
Knob Circuit

For the **Knob** example, wire the potentiometer so that its two outer pins are connected to power (+5V) and ground, and its middle pin is connected to A0 on the board. Then, connect the servo motor as shown in the circuit below.



Sweep Circuit

For the **Sweep** example, connect the servo motor as shown in the circuit below.



Examples

Knob

Controlling a servo position using a potentiometer (variable resistor).

```
#include <Servo.h>
Servo myservo;  // create servo object to control a servo
int potpin = 0;  // analog pin used to connect the potentiometer
int val;    // variable to read the value from the analog pin

void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}
void loop() {
  val = analogRead(potpin);            // reads the value of the potentiometer (value 
between 0 and 1023)
  val = map(val, 0, 1023, 0, 180);     // scale it to use it with the servo (value 
between 0 and 180)
  myservo.write(val);                  // sets the servo position according to the 
scaled value
  delay(15);                           // waits for the servo to get there
}
```

Sweep

Sweeps the shaft of a RC servo motor back and forth across 180 degrees.

```
#include <Servo.h>
Servo myservo;  // create servo object to control a servo
// twelve servo objects can be created on most boards
int pos = 0;    // variable to store the servo position

void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}
void loop() {
  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 
'pos'
    delay(15);                       // waits 15ms for the servo to reach the 
position
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in variable 
'pos'
    delay(15);                       // waits 15ms for the servo to reach the 
position
  }
}
```