
# Arduino Robotics Kit

Document No. ODD-007 January 2014

# Contents

# Overview

The oddWires Arduino Robotics Kit is designed to be built and used by individuals, educational institutions and anyone who wants to learn about robotics. It offers fantastic value for money including a complete, extensible chassis with an Arduino Uno to control and manage the robot.

The kit includes both breadboarding and soldering options via a 400 tie-point breadboard and a prototype board with stackable headers. So you can build it without soldering first and after testing you can solder it.

The robot itself consists of a chassis onto which you will mount an Arduino Uno together with a stackable Arduino shield that you will wire up from a generic prototype board. This prototype board will have terminals for the two motors and a terminal for the 6V battery holder that will supply the power for the motors and the servo (the Uno will be supplied by a separate 9V power supply). It will also have the L393D chip that controls the motors as well as a connector for the servo. The connections from the ultrasonic distance sensor will be made to the stackable headers.

The kit includes more than strictly necessary to build the robot so that you use it to learn much more than basic robotics. It includes LEDs so can can quickly get to grips with simple Arduino sketches to blink LEDs (and learn ohms law in the process).

You can also use the supplied MOSFETs to gain an introduction to switching higher power circuits from the Arduino.

That's followed by an introduction to the L293D motor controller IC and H-bridge circuits.

Once the basic motor controller is functioning, you can then add a servo-mounted ultrasonic distance sensor to enable the robot to maneuver using its own logic.

Sketches are supplied at several levels from introductory to a complete robot controller. All of these sketches are extensible and you can add all sorts of additional sensors and enhance the initial robot behavior.

## Robot Construction Steps

- Check your kit contents list and ensure you have all components.
- Install the Arduino IDE and, if necessary USB drivers.
- (Optional) Breadboard a LED with current-limiting resistor, see the effects of ohms law by varying the current limiting resistor, and blink the LED from an Arduino sketch.
- (Optional) Use one or more MOSFETs to control one or more motors from the Arduino.
- (Optional) Breadboard the L293D motor controller IC with the Arduino to see how to control both motors with forward, reverse and braking. Understand how and H-bridge motor controller works.
- (Optional) Learn to use the servo before using it on the chassis to rotate the ultrasonic distance sensor
- (Optional) Learn to use the ultrasonic distance sensor before using it on the robot chassis
- Construct the chassis and the 6V power supply for the driving motors and servo.
- Install the Arduino Uno and its 9V power supply
- Construct the prototype board, motor control first, then the servo, and lastly the ultrasonic distance sensor (there are separate sketches to test each step). Install it via the stackable headers.
- Load the final sketch and test your completed robot.

# Kit Contents

| **Chassis** | Quantity |
|---|---|
| Paper-wrapped Acrylic Chassis Base | 1 |
| Motors with Leads | 2 |
| Wheels | 2 |
| Castor Wheel | 1 |
| 4 x AA Battery Case for Motor Drive | 1 |
| Hardware Pack | 1 |
| **Arduino Uno** | |
| Arduino Uno R3 | 1 |
| USB cable | 1 |
| 9V Battery Case with 2.1mm Plug for Arduino | 1 |
| **Motor Driver Board** | |
| Prototype Board | 1 |
| L293D Motor Driver IC | 1 |
| 16 Pin IC Socket | 1 |
| Connecting Wire (4 colors) | 4 |
| Stackable Headers (2 x 6 Pin, 2 x 8 Pin), Set | 1 |
| 2 Position Terminal Blocks (Left Motor, Right Motor & Motor Drive Battery Case) | 3 |
| **Servo** | |
| Servo with Lead and Accessories | 1 |
| | 1 |
| **Ultrasonic Sensor** | |
| Ultrasonic Distance Sensor Module HC-SR04 | 1 |
| 1x 4 way ribbon cable M-F | 1 |
| **Misc.** | |
| Double-sided tape for attaching ultrasonic module to servo arm, breadboard and 9V battery case to chassis | 1 |

None of the following items are required to construct the robotics kit but are provided to enhance the educational value of the kit. For example, one might breadboard a simple power drive solution using the MOSFETs. Alternatively, you could breadboard the motor drive before soldering the prototype board. Various LEDs and resistors are included for driving indicators or providing status.

| | |
|---|---|
| **LED Pack** | |
| 3 mm red LEDs | 3 |
| 3 mm yellow LEDs | 3 |
| 3 mm green LEDs | 3 |
| 5 mm red LEDs | 3 |
| 5 mm green LEDs | 3 |
| **Resistors** | |
| 220 ohm resistors | 10 |
| 10K resistors | 10 |
| 1K resistors | 10 |
| **Misc.** | |
| Ribbon cable M-M | 1 |
| 400 point breadboard for rapid prototyping without soldering | 1 |
| MOSFET IRLB8721PBF | 2 |

# Hardware Pack Contents

| **Part List** | Quantity |
|---|---|
| Motor Mounts | 2 |
| Rotors | 2 |
| M3 x 30mm Screws, Pan-Headed | 4 |
| M3 x 6mm Screws, Flat-Headed | 4 |
| M3 x 8mm Screws, Pan-Headed | 6 |
| M3 Nuts | 8 |
| M3 x 10mm Female-Female Stand-Offs | 4 |
| M3 x 25mm Female-Female Stand-Offs | 2 |
| M3 x 15mm Female-Female Stand-Offs | 4 |
| M3 x 6mm Screws, Pan-Headed | 14 |

These parts are used as follows:

**Castor Wheel**

| | |
|---|---|
| M3 x 15mm Female-Female Stand-Offs | 4 |
| M3 x 6mm Screws, Flat-Headed | 4 |
| M3 x 8mm Screws, Pan-Headed | 4 |

**Arduino Uno**

| | |
|---|---|
| M3 x 10mm Female-Female Stand-Offs | 4 |
| M3 x 6mm Screws, Pan-Headed (only two stand-offs are screwed to chassis base) | 6 |

**Motors**

| | |
|---|---|
| Motor Mounts | 2 |
| Rotors | 2 |
| M3 x 30mm Screws, Pan-Headed | 4 |
| M3 x 6mm Screws, Pan-Headed | 4 |
| M3 Nuts | 4 |

**Servo**

| | |
|---|---|
| M3 x 25mm Female-Female Stand-Offs | 2 |
| M3 x 8mm Screws, Pan-Headed | 2 |
| M3 x 6mm Screws, Pan-Headed | 2 |
| M3 Nuts | 2 |

**6V 4 x AA Battery Case**

| | |
|---|---|
| M3 x 6mm Screws, Pan-Headed | 2 |
| M3 Nuts | 2 |

**Note:** hardware may be in one pack or the following may be in a separate pack:

| | |
|---|---|
| M3 x 25mm Female-Female Stand-Offs | 2 |
| M3 x 15mm Female-Female Stand-Offs | 4 |
| M3 x 6mm Screws, Pan-Headed | 14 |

**Tools and materials required for assembly**

- Soldering iron
- Solder
- Pliers
- Wire snips & wire stripper
- Cross-Head Screwdriver

# Install the Arduino IDE and USB Drivers



## Getting Started with Arduino on Windows

### Download the Arduino environment

Get the latest version from the download page (http://arduino.cc/en/Main/Software).

When the download finishes, unzip the downloaded file. Make sure to preserve the folder structure. Double-click the folder to open it. There should be a few files and sub-folders inside.

### Connect the board

The Arduino Uno automatically draws power from either the USB connection to the computer or an external power supply. If you're using an Arduino Diecimila, you'll need to make sure that the board is configured to draw power from the USB connection. The power source is selected with a jumper, a small piece of plastic that fits onto two of the three pins between the USB and power jacks. Check that it's on the two pins closest to the USB port.

Connect the Arduino board to your computer using the USB cable. The green power LED (labeled **PWR**) should light.

### Install the drivers

**Installing drivers for the Arduino Uno with Windows8, Windows7, Vista, or XP:**

- Plug in your board and wait for Windows to begin its driver installation process. After a few moments, the process will fail, despite its best efforts
- Click on the Start Menu, and open up the Control Panel.
- While in the Control Panel, navigate to System and Security. Next, click on System. Once the System window is up, open the Device Manager.
- Look under Ports (COM & LPT). You should see an open port named "Arduino UNO (COMxx)"
- Right click on the "Arduino UNO (COMxx)" port and choose the "Update Driver Software" option.
- Next, choose the "Browse my computer for Driver software" option.
- Finally, navigate to and select the driver file named **"arduino.inf"**, located in the "Drivers" folder of the Arduino Software download (not the "FTDI USB Drivers" sub-directory). If you are using an old version of the IDE (1.0.3 or older), choose the Uno's driver file named **"Arduino UNO.inf"**
- Windows will finish up the driver installation from there.

When you connect the board, Windows should initiate the driver installation process (if you haven't used the computer with an Arduino board before).

On Windows 8, 7 or Vista, the driver should be automatically downloaded and installed. You can check that the drivers have been installed by opening the Windows Device Manager (in the Hardware tab of System control panel). Look for a "USB Serial Port" in the Ports section; that's the Arduino board.

### Launch the Arduino application

Double-click the Arduino application. (Note: if the Arduino software loads in the wrong language, you can change it in the preferences dialog. See the environment page (http://arduino.cc/en/Guide/Environment#languages) for details.)

## Getting Started with Arduino on Mac

### Download the Arduino environment

Get the latest version from the download page (http://arduino.cc/en/Main/Software). When the download is finished, double click the .zip file. This will expand the Arduino application.

### Install the Software

Copy the Arduino application into the Applications folder (or elsewhere on your computer). No drivers are required to be installed.

### Connect the board

The Arduino Uno automatically draws power from either the USB connection to the computer or an external power supply. Connect the Arduino board to your computer using the USB cable. The green power LED (labeled **PWR**) should go on.

A dialog box will appear telling you that a new network interface has been detected. Click "Network Preferences...", and when it opens, simply click "Apply". The Uno or Mega 2560 will show up as "Not Configured", but it's working properly. Quit System Preferences.

### Launch the Arduino application

Double-click the Arduino application. **Note:** if the Arduino software loads in the wrong language, you can change it in the preferences dialog. See the environment page for details (http://arduino.cc/en/Guide/Environment#languages)

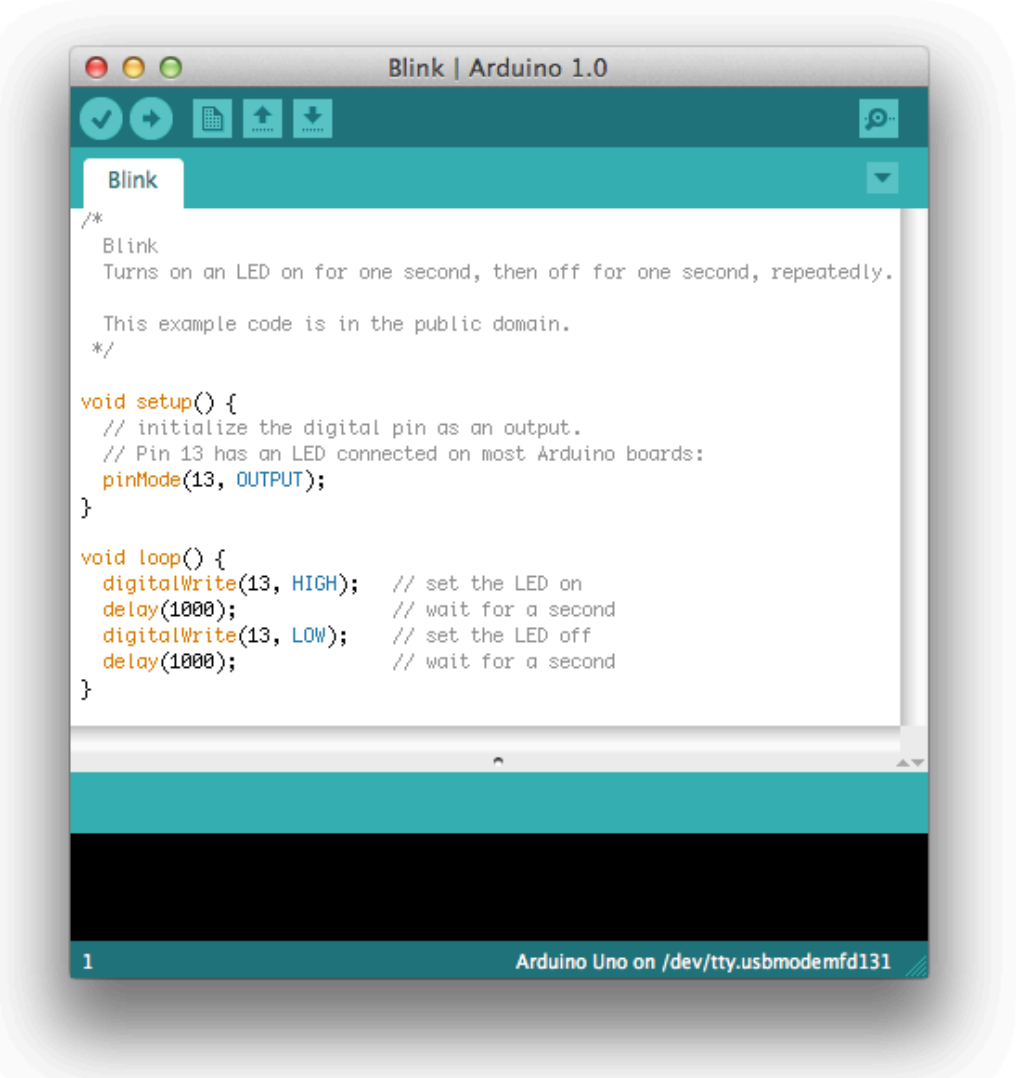
## Getting Started with Arduino on Linux

Getting Started on Linux depends on your particular distribution. Details can be found here (http://playground.arduino.cc/Learning/Linux).

# Your First Arduino Sketch

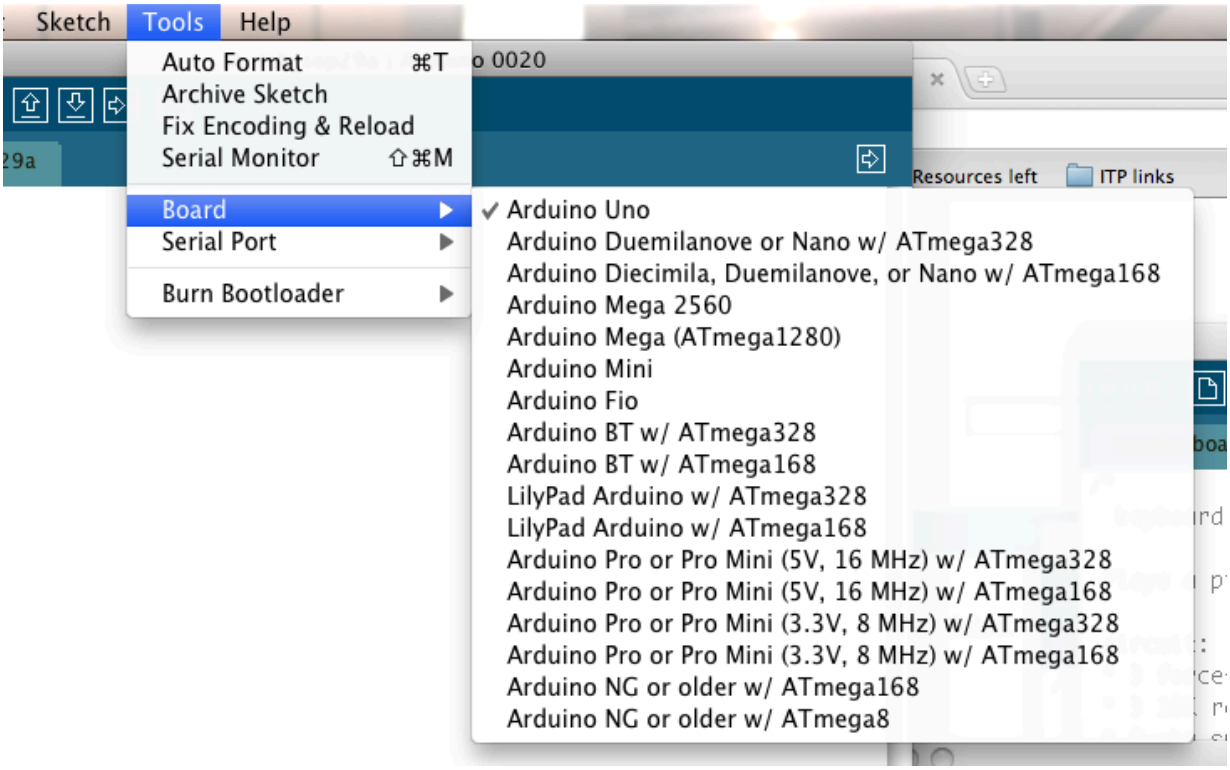
## Open the blink example

Open the LED blink example sketch: File > Examples > 1.Basics > Blink.




## Select your board

You'll need to select the entry in the **Tools > Board** menu that corresponds to your Arduino.



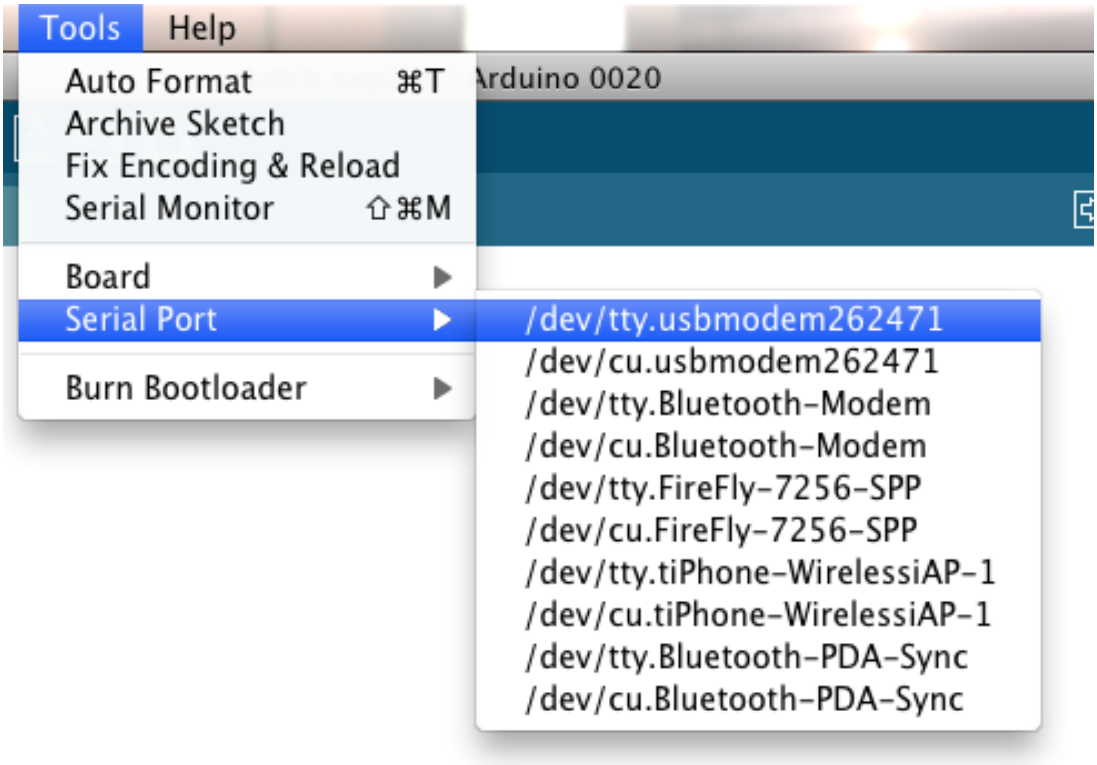

Selecting an Arduino Uno

## Select your serial port

Select the serial device of the Arduino board from the **Tools > Serial Port** menu.

Select the serial device of the Arduino board from the Tools | Serial Port menu. On Windows this is likely to be **COM3** or higher (**COM1** and **COM2** are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu; the entry that disappears should be the Arduino board. Reconnect the board and select that serial port.

On the Mac, this should be something with **/dev/tty.usbmodem** (for the Uno).




Selecting an Uno

## Upload the program

Now, simply click the "Upload" button in the environment. Wait a few seconds - you should see the RX and TX LEDs on the board flashing. If the upload is successful, the message "Done uploading." will appear in the status bar.


A few seconds after the upload finishes, you should see the pin 13 (L) LED on the board start to blink. Congratulations! You've got Arduino up-and-running.

If you have problems, please see the troubleshooting suggestions http://arduino.cc/en/Guide/Troubleshooting).

# Blink

This example shows the simplest thing you can do with an Arduino to see physical output: it blinks an LED.

## Hardware Required

- Arduino Board
- LED

## Circuit

To build the circuit, attach a 220-ohm resistor to pin 13. Then attach the long leg of an LED (the positive leg, called the anode) to the resistor. Attach the short leg (the negative leg, called the cathode) to ground. Then plug your Arduino board into your computer, start the Arduino program, and enter the code below.

Most Arduino boards already have an LED attached to pin 13 on the board itself. If you run this example with no hardware attached, you should see that LED blink.

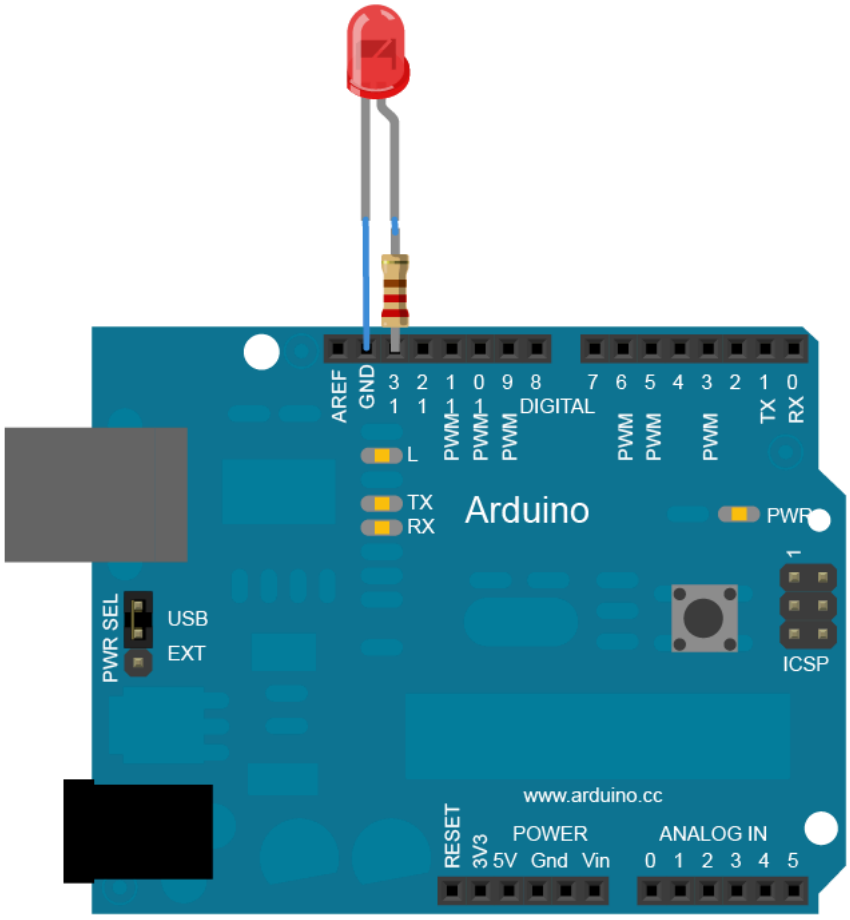
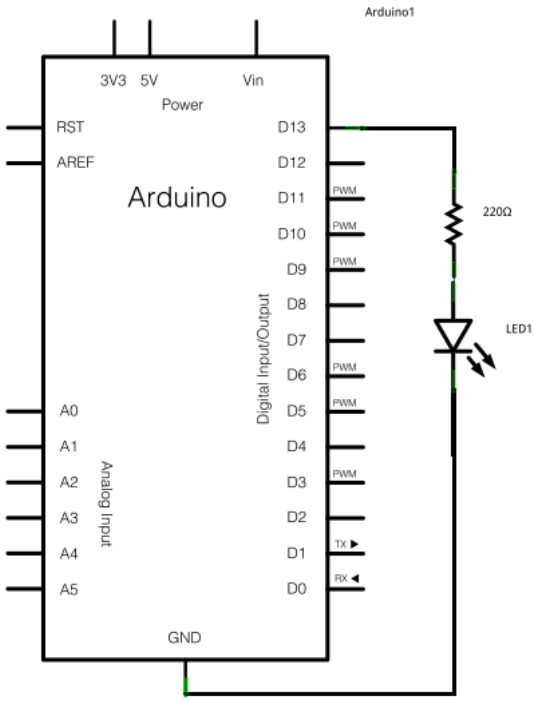



Image developed using Fritzing. For more circuit examples, see the Fritzing project page

## Schematic

Click the image to enlarge




## Code

In the program below, the first thing you do is to initialize pin 13 as an output pin with the line

```
pinMode(13, OUTPUT);
```

In the main loop, you turn the LED on with the line:

```
digitalWrite(13, HIGH);
```

This supplies 5 volts to pin 13. That creates a voltage difference across the pins of the LED, and lights it up. Then you turn it off with the line:

```
digitalWrite(13, LOW);
```

That takes pin 13 back to 0 volts, and turns the LED off. In between the on and the off, you want enough time for a person to see the change, so the `delay()` commands tell the Arduino to do nothing for 1000 milliseconds, or one second. When you use the `delay()` command, nothing else happens for that amount of time. Once you've understood the basic examples, check out the BlinkWithoutDelay example to learn how to create a delay while doing other things.

Once you've understood this example, check out the DigitalReadSerial example to learn how read a switch connected to the Arduino.

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// Give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

# Ohm's Law

So why did we use a 220 ohm resistor in series with the LED? This is because the maximum current that the Arduino pins are able to sink or source is 40ma. The sum total for all the pins simultaneously should not be more than 200ma.

So how much current will flow though a 220 ohm resistor with a 5V potential difference - 5V to 0V (GND).

Ohm's law says V = IR or in this case we want to find I = V/R. So I = 5/220 = 23ma. This is well within the allowable current for the Arduino pin.

If you change the current limiting resistor to 1K ohm you will get a current around 5ma. This may not be enough to light the LED or it will appear very dim.

So the correct choice of resistor is a function of the maximum current that the Arduino pin can handle and the current required to light the LED.

}

# Fading a LED using PWM

Demonstrates the use of the analogWrite() function in fading an LED off and on. `AnalogWrite` uses pulse width modulation (PWM), turning a digital pin on and off very quickly, to create a fading effect. We will use this same PWM technique to vary the speed of the DC motors in our robot.

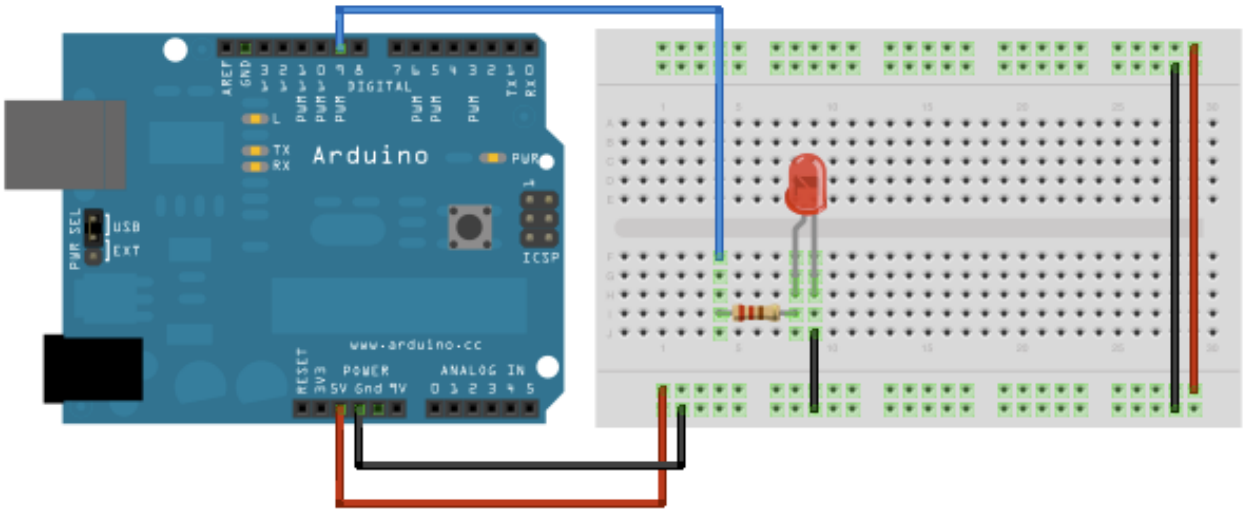
## Hardware Required

- Arduino board
- Breadboard
- a LED
- a 220 ohm resistor

## Circuit

Connect the **anode** (the longer, positive leg) of your LED to digital output pin 9 on your Arduino through a 220-ohm resistor. Connect the **cathode** (the shorter, negative leg) directly to ground.

Click the image to enlarge



# Schematic

Click the image to enlarge

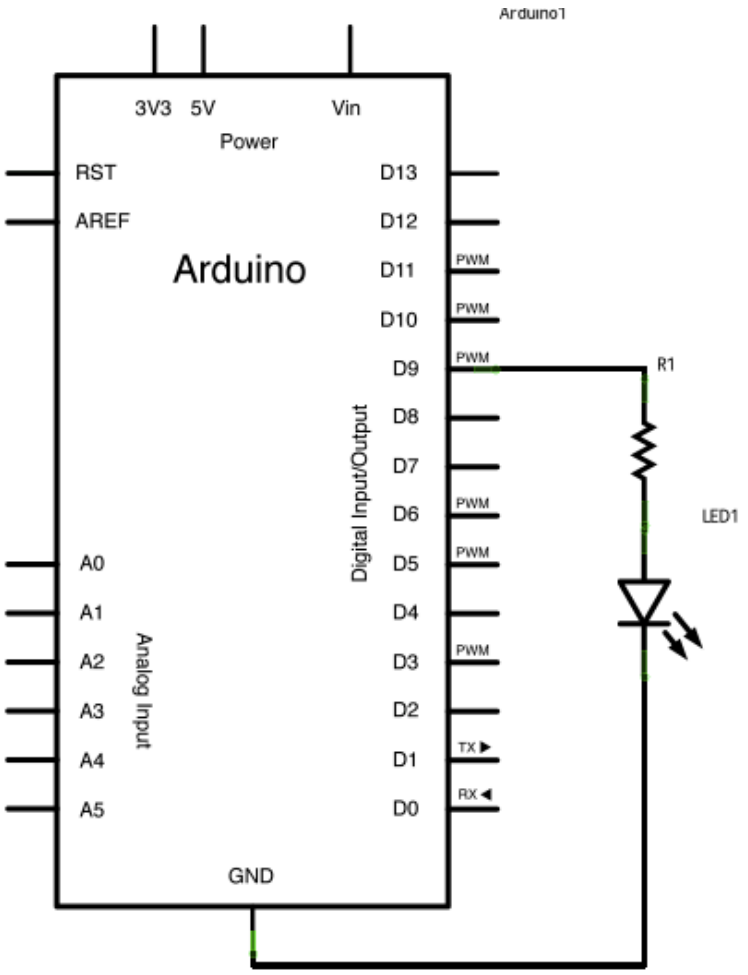



Image developed using Fritzing. For more circuit examples, see the Fritzing project page

# Code

After declaring pin 9 to be your ledPin, there is nothing to do in the `setup()` function of your code.

The `analogWrite()` function that you will be using in the main loop of your code requires two arguments: One telling the function which pin to write to, and one indicating what PWM value to write.

In order to fade your LED off and on, gradually increase your PWM value from 0 (all the way off) to 255 (all the way on), and then back to 0 once again to complete the cycle. In the sketch below, the PWM value is set using a variable called `brightness`. Each time through the loop, it increases by the value of the variable `fadeAmount`.

If `brightness` is at either extreme of its value (either 0 or 255), then `fadeAmount` is changed to its negative. In other words, if `fadeAmount` is 5, then it is set to -5. If it's 55, then it's set to 5. The next time through the loop, this change causes `brightness` to change direction as well.

`analogWrite()` can change the PWM value very fast, so the delay at the end of the sketch controls the speed of the fade. Try changing the value of the delay and see how it changes the program.

```
/*
 Fade

 This example shows how to fade an LED on pin 9
 using the analogWrite() function.

 This example code is in the public domain.
 */

int led = 9;           // the pin that the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by

// the setup routine runs once when you press reset:
void setup()  {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop()  {
  // set the brightness of pin 9:
  analogWrite(led, brightness);

  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness == 0 || brightness == 255) {
    fadeAmount = -fadeAmount ;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

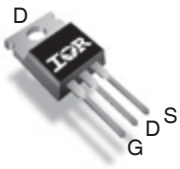
# Controlling a DC Motor with a MOSFET

We have learnt already that there is a limit of 40ma for an Arduino pin. So how can we control circuits that require larger currents such as motors or even mains circuits? There are three key ways of doing this and they each have their own advantages and disadvantages:

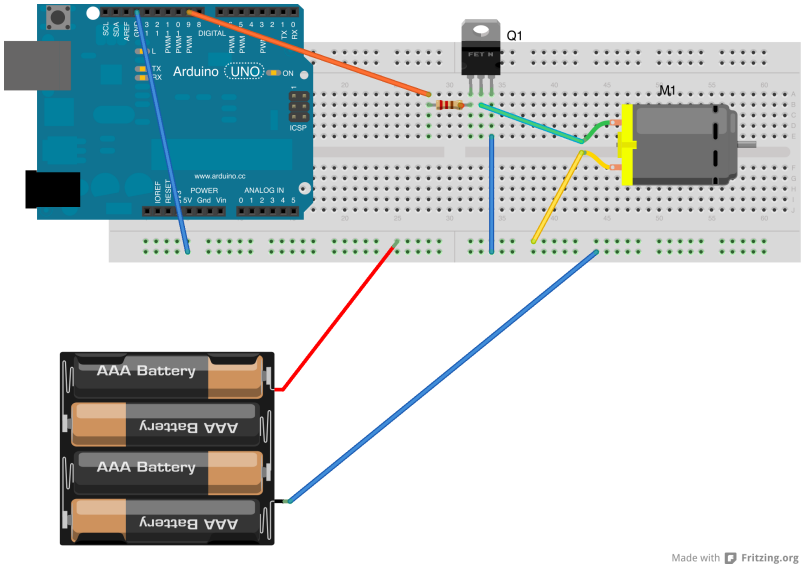
1. Relays – mechanical or solid-state. This is the easiest way of controlling mains circuits – where simple on/off control is required. Solid-sate relays are typically used where safe operating conditions preclude mechanical action that may create sparks. Opto-isolators are often used to further isolate the mains circuit from the low-level circuit.
2. Where mains circuits are being varied (for example a light dimmer) then a triac can be used often in conjunction with an opto-isolator. They can be used for on-off switching too.
3. The simplest circuit for motor control is the use of a transistor. It's possible to use a transistor (BJT or a MOSFET) but the MOSFET applies infinitesimal load on the driving circuit so it has a clear advantage.

## Using the supplied MOSFETs to drive a DC motor

Without going into the complexities of the MOSFET we can treat it as a switch. If we apply a potential to the Gate it will switch the current through the Source/Drain junction. So we connect the gate to the Arduino through a 220 ohm current limiting resistor to protect the Arduino. Then we connect the motor to the external power supply and the Drain of the MOSFET. We complete the circuit by connecting up the grounds. In this case the MOSFET controlling the motor is driven from Arduino Pin 9.




TO-220AB
IRLB8721PbF



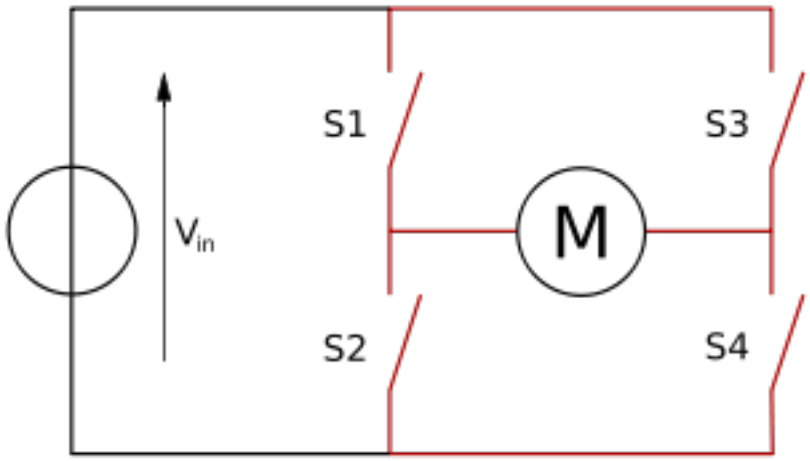
```
/*
Speed up the motor
This example shows how to control the speed of a DC motor an LED on pin 9 using the
analogWrite() function. This example based on the Arduino Example Fade sketch but
modified to use timing instead of the delay() function
*/
int turns = 0; // how fast the motor runs
int turnAmount = 1; // how many turns the motor makes
unsigned long currentTime;
unsigned long loopTime;

void setup() {
// declare pin 9 to be an output:
    pinMode(9, OUTPUT);
    currentTime = millis();
    loopTime = currentTime;
}

void loop() {
    currentTime = millis();
    if(currentTime >= (loopTime + 20)){
        // set the speed of pin 9:
        analogWrite(9, turns);

        // change the turnings for next time through the loop:
        turns = turns + turnAmount;

        // speed up or slow down the motor
        if (turns == 0 || turns == 255) {
            turnAmount = -turnAmount ;
        }
        if (turns == 0) {
            delay(5000);
        }
        loopTime = currentTime; // Updates loopTime
    }
    // Other processing can be done here
}
```

# Using the L293D Motor Controller

In this section we will look at using an H-bridge to control a motor and then how we use an L293D quad half-H bridge to control two motors.

## Using an H bridge

The H-bridge arrangement is used to reverse the polarity of the motor and can also be used to 'brake' the motor, where the motor comes to a sudden stop, as the motor's terminals are shorted, or to let the motor 'free run' to a stop, as the motor is effectively disconnected from the circuit. The following table summarizes operation, with S1-S4 corresponding to the diagram below.

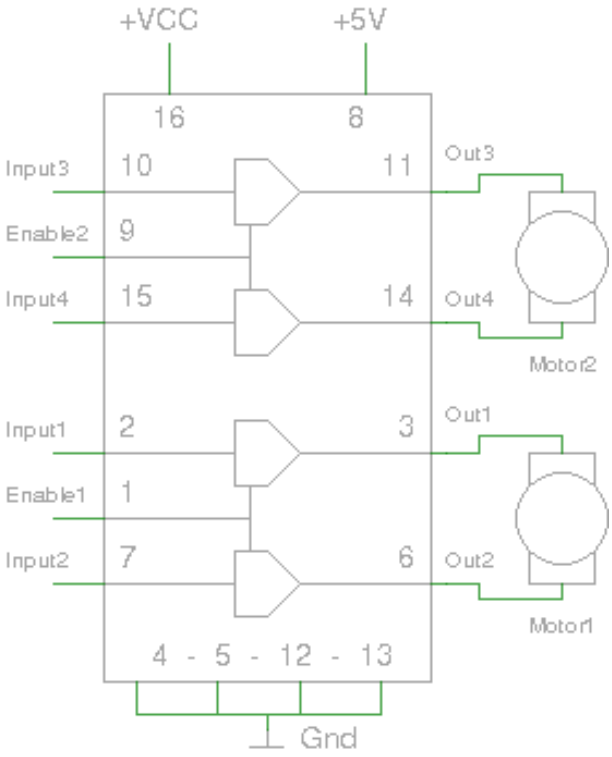



| S1 | S2 | S3 | S4 | Result |
|---|---|---|---|---|
| 1 | 0 | 0 | 1 | Motor moves right |
| 0 | 1 | 1 | 0 | Motor moves left |
| 0 | 0 | 0 | 0 | Motor free runs |
| 0 | 1 | 0 | 1 | Motor brakes |
| 1 | 0 | 1 | 0 | Motor brakes |
| 1 | 1 | 0 | 0 | Shoot-through (short-circuit) |
| 0 | 0 | 1 | 1 | Shoot-through (short-circuit) |
| 1 | 1 | 1 | 1 | Shoot-through (short-circuit) |

## The L293D

The L293 and L293D are quadruple high-current half-H drivers. The L293 is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications. Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN. When an enable input is high, the associated drivers are enabled, and their outputs are active and in phase with their inputs. When the enable input is low, those drivers are disabled, and their outputs are. With the proper data inputs, each pair of drivers forms a full-H (or bridge) reversible drive suitable for solenoid or motor applications. **Note:** we are using the L293D because it simplifies the circuit by having fly-back diodes built-in. These prevent unwanted voltage spikes caused when the motors are switched on or off.

Here are the connections required. Note the separate power supply on pin 16 from the logic power supply on pin 8. The enable pins 1 and 9 can be controlled using PWM to set the motor speed. The input pins control direction and braking. Look at the Arduino Sketch for a truth table of what happens for each pin combination.



# Sweeping the Servo by 180 Degrees

In this section we learn how to use a servo with Arduino. We will sweep the shaft of a RC servomotor back and forth across 180 degrees. In this example the servo is powered from the Arduino. When we build our robot we will power the servo from a separate battery because it may draw too much power from the Arduino.

This example makes use of the Arduino **servo library**.

## Hardware Required

- Arduino Board
- (1) Servo Motor
- Jumper wire

## Circuit

Servo motors have three wires: power, ground, and signal. The power wire is typically red, and should be connected to the 5V pin on the Arduino board. The ground wire is typically black or brown and should be connected to a ground pin on the Arduino board. The signal pin is typically yellow, orange or white and should be connected to pin 9 on the Arduino board.

Click the image to enlarge

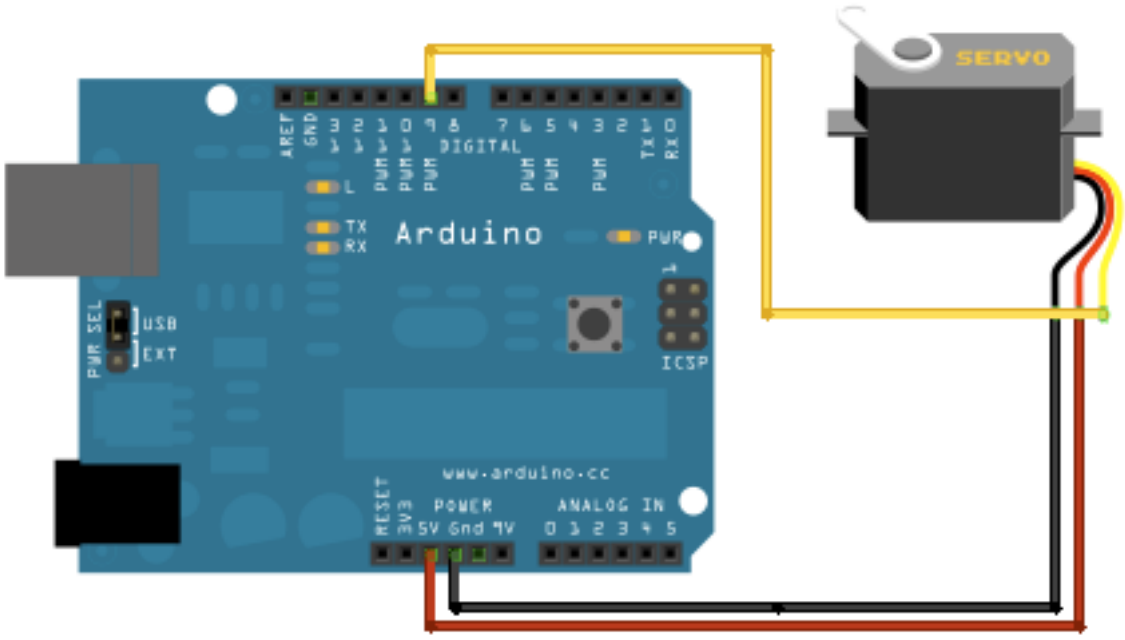
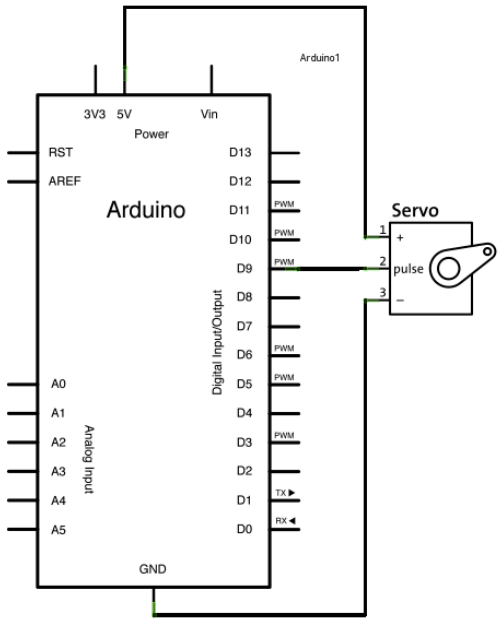



Image developed using Fritzing. For more circuit examples, see the Fritzing project page

## Schematic




## Code

```
// Sweep
// by BARRAGAN <http://barraganstudio.com>
// this example code is in the public domain.


#include <Servo.h>

Servo myservo;  // create servo object to control a servo
                // a maximum of eight servo objects can be created

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>=1; 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
  }
}
```

# HC-SR04 Ultrasonic Distance Sensor

The HC-SR04 is an ultrasonic distance sensor. It detects the distance of the closest object in front of the sensor (from 2 cm up to 3m). It works by sending out a burst of ultrasound and listening for the echo when it bounces off of an object. The Arduino board sends a short pulse to trigger the detection, then listens for a pulse on the echo pin. The duration of this second pulse is equal to the time taken by the ultrasound to travel to the object and back to the sensor. Using the speed of sound, this time can be converted to distance.

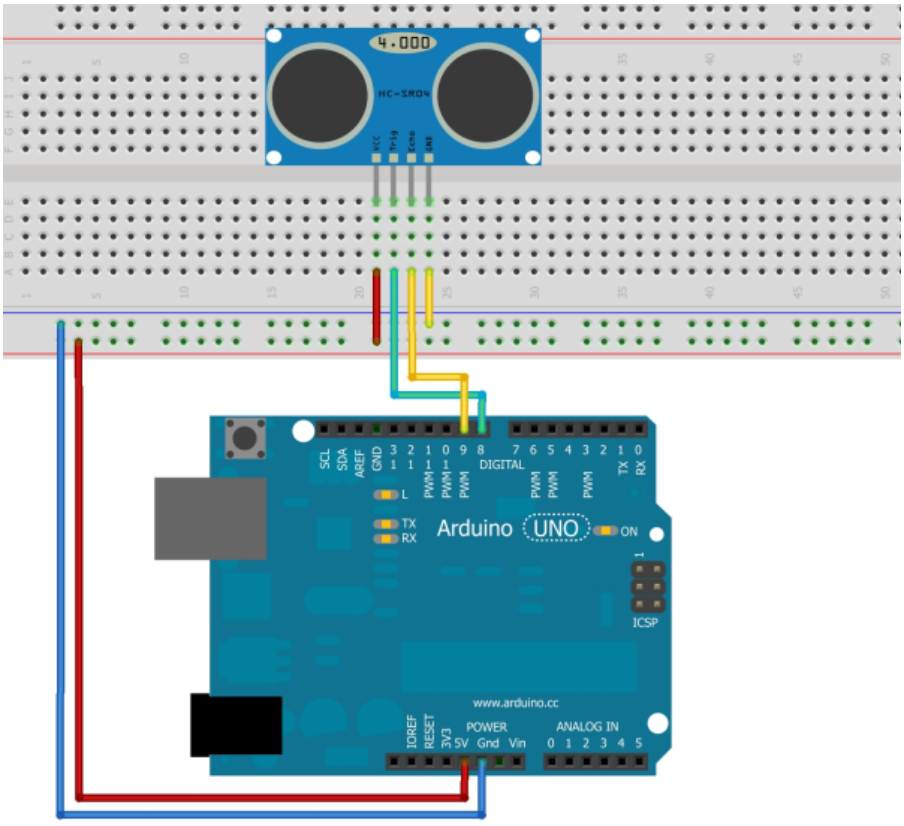
## Hardware Required

- Arduino Board
- (1) HC-SR04 Ultrasonic Distance Sensor
- Jumper wire

### *Circuit*

The 5V pin of the PING))) is connected to the 5V pin on the Arduino, the GND pin is connected to the GND pin, and the SIG (signal) pin is connected to digital pin 7 on the Arduino.

Image developed using Fritzing. For more circuit examples, see the Fritzing project page



# Code

```
/*
HC-SR04 for Arduino

Original project from http://www.swanrobotics.com

This project demonstrates the HC-SR
The distance presented in the code is in mm, but you can uncomment the line
for distance in inches.
The schematics for this project can be found on http://www.swanrobotics.com

This example code is in the public domain.
*/

const int TriggerPin = 8;      //Trig pin
const int EchoPin = 9;         //Echo pin
long Duration = 0;

void setup(){
  pinMode(TriggerPin,OUTPUT);  // Trigger is an output pin
  pinMode(EchoPin,INPUT);      // Echo is an input pin
  Serial.begin(9600);          // Serial Output
}

void loop(){
  digitalWrite(TriggerPin, LOW);
  delayMicroseconds(2);
  digitalWrite(TriggerPin, HIGH);          // Trigger pin to HIGH
  delayMicroseconds(10);                   // 10us high
  digitalWrite(TriggerPin, LOW);           // Trigger pin to HIGH

  Duration = pulseIn(EchoPin,HIGH);        // Waits for the echo pin to get
high
                                           // returns the Duration in
microseconds
  long Distance_mm = Distance(Duration);   // Use function to calculate the
distance

  Serial.print("Distance = ");             // Output to serial
  Serial.print(Distance_mm);
  Serial.println(" mm");

  delay(1000);                             // Wait to do next measurement
}

long Distance(long time)
{
    // Calculates the Distance in mm
    // ((time)*(Speed of sound))/ toward and backward of object) * 10

    long DistanceCalc;                      // Calculation variable
    DistanceCalc = ((time /2.9) / 2);     // Actual calculation in mm
    //DistanceCalc = time / 74 / 2;         // Actual calculation in inches
    return DistanceCalc;                    // return calculated value
}
```

# Constructing the Chassis

Identify the chassis components:

1 x chassis plate

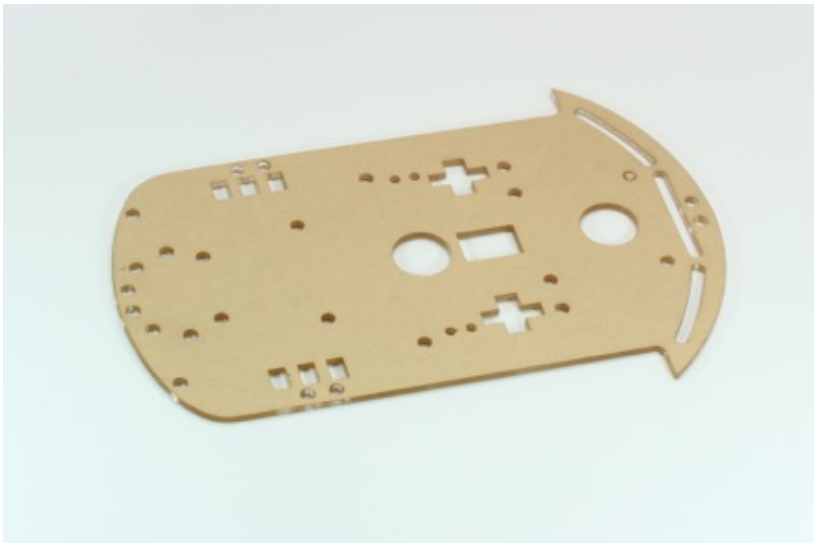
2 x wheels

2 x motors

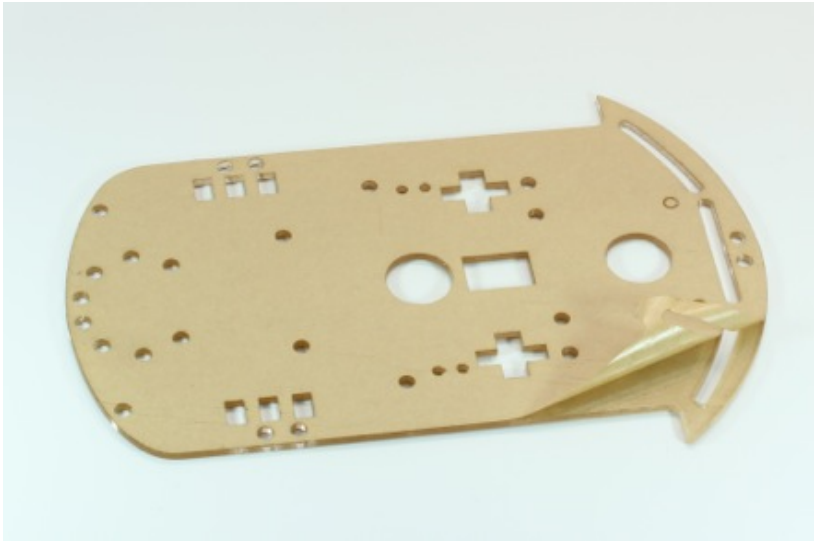
1 x castor wheel

1 x hardware pack

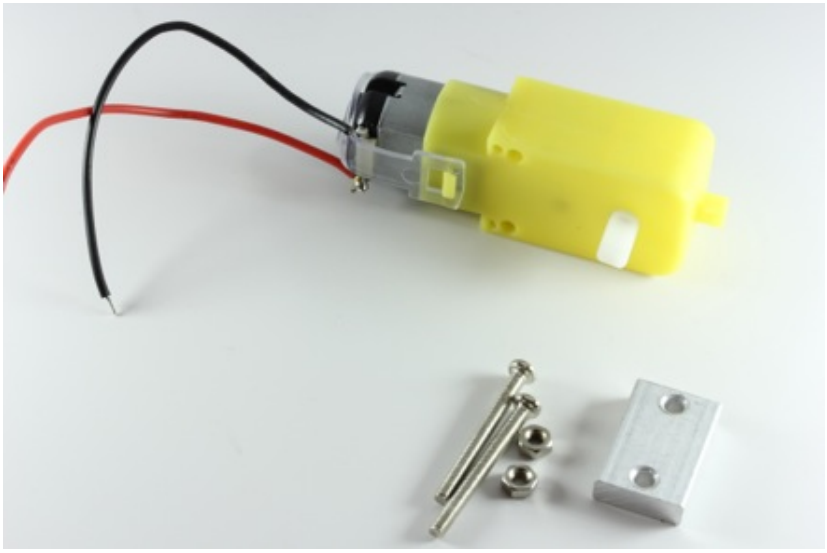
Here's the chassis plate



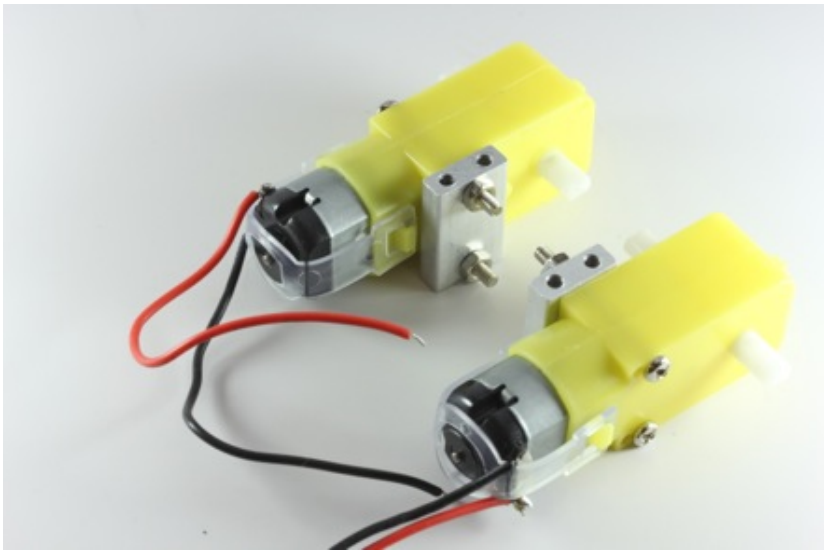
Remove the layer of protective paper from both sides.



Here's the motor, mounting plate and the two long machine screws & two nuts.



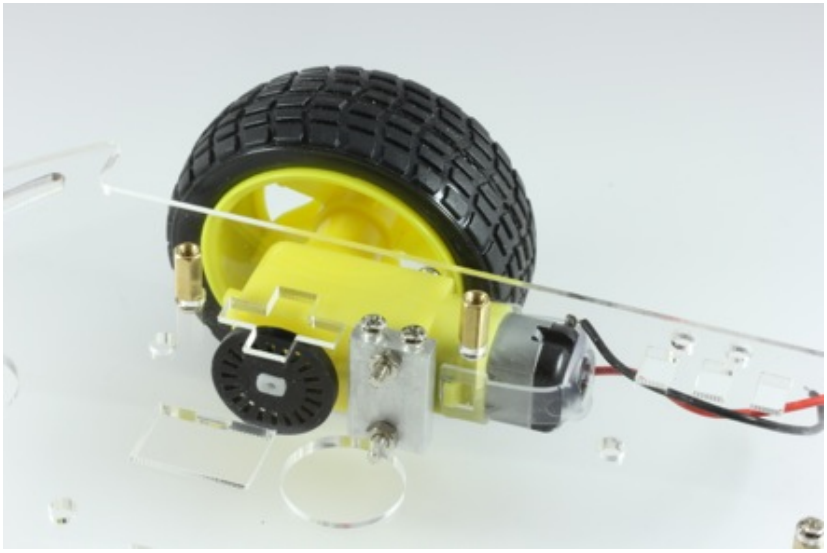
Both of the mounts attached (note mirror image).



Attach the motors to the chassis with two pan-head screws.

Tighten them up after you have threaded both screws. It will be hard to align the screws if you tighten up one first.

Push on the wheels and rotors.



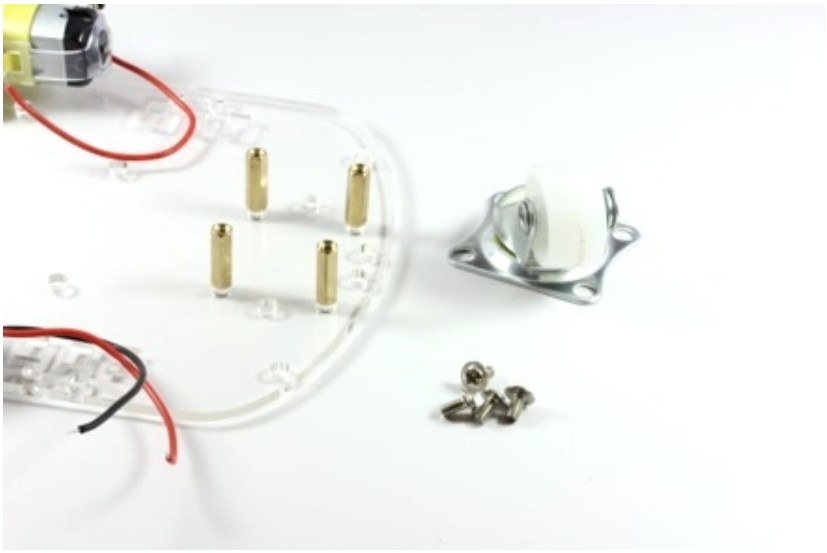
Mounting the castor wheel requires:

4 x 15 mm standoffs

4 flat-head screws

4 x pan-head screws

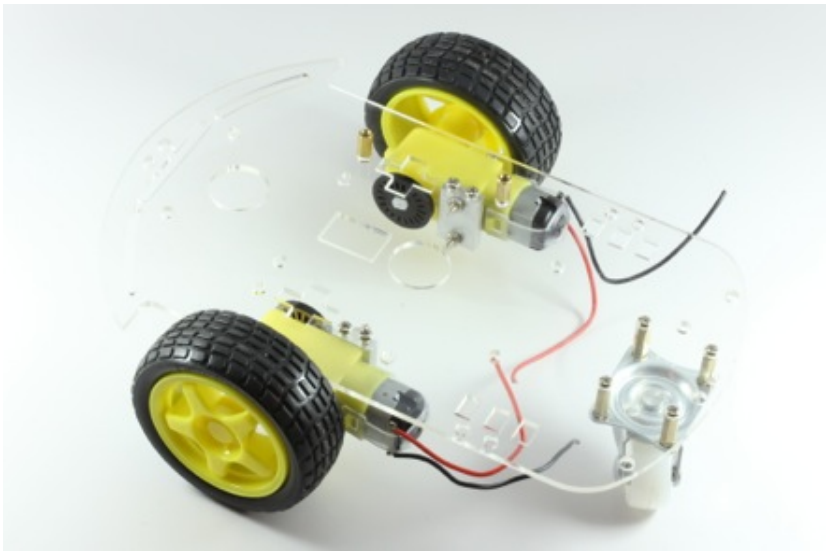
Screw the stand-offs to the chassis with the pan-head screws.



Mount the castor wheel with the flat-head screws.

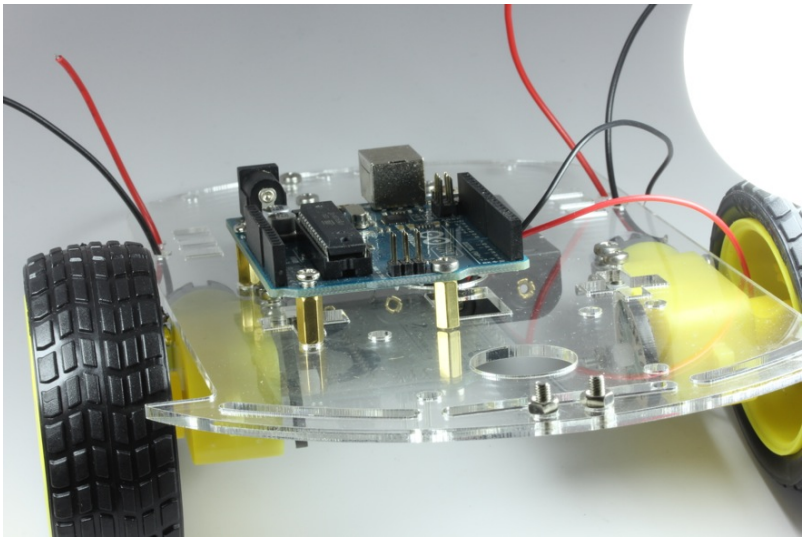
Here is the assembled chassis.

Another view.

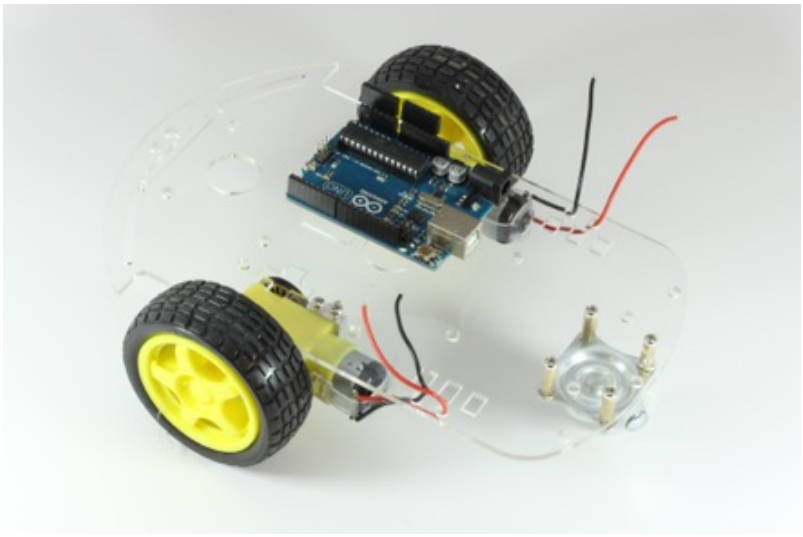


# Installing the Arduino Uno

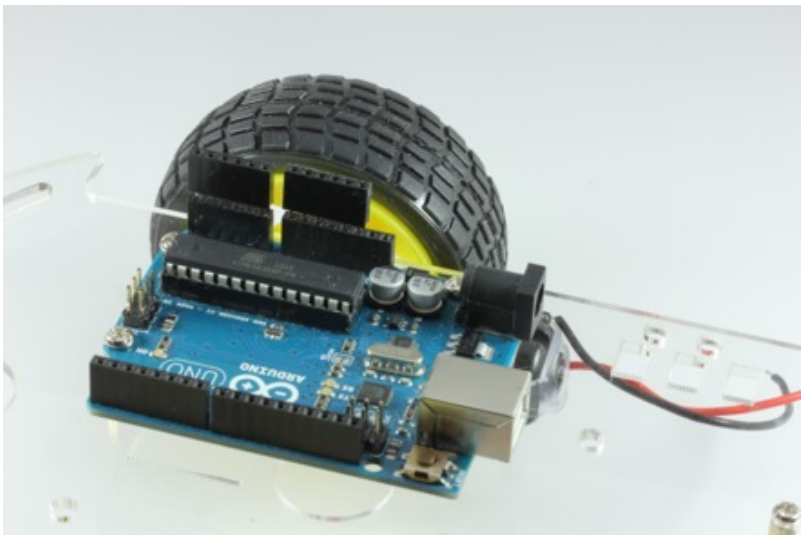
The Uno is mounted with the 4 smaller standoffs. Two are simply supports and two are screwed to the chassis. Use 6 pan-head screws in total.



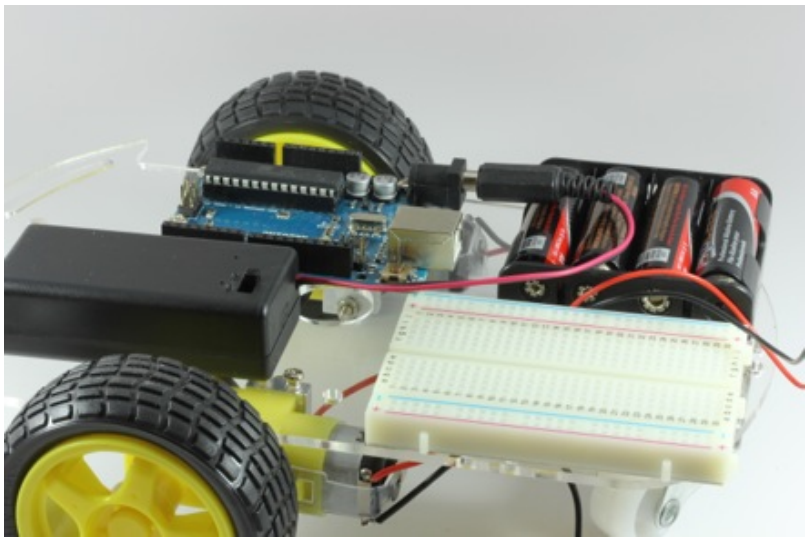
This is where the Uno is mounted.



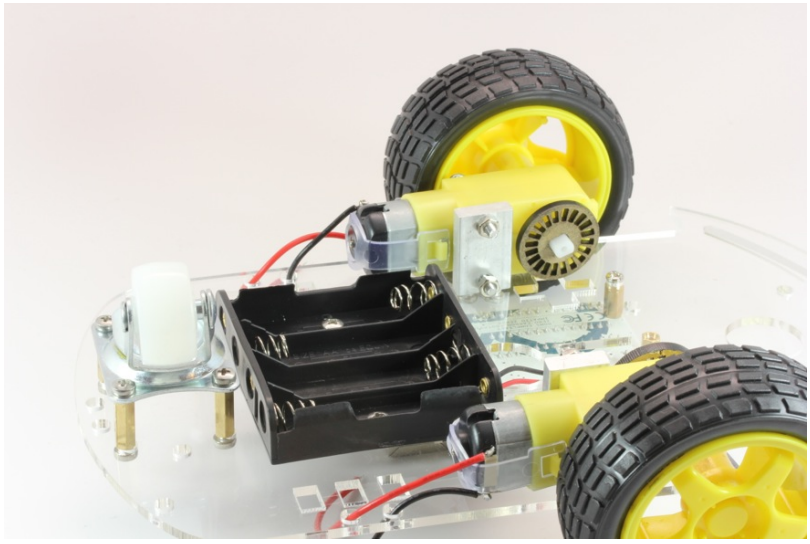
Another view.



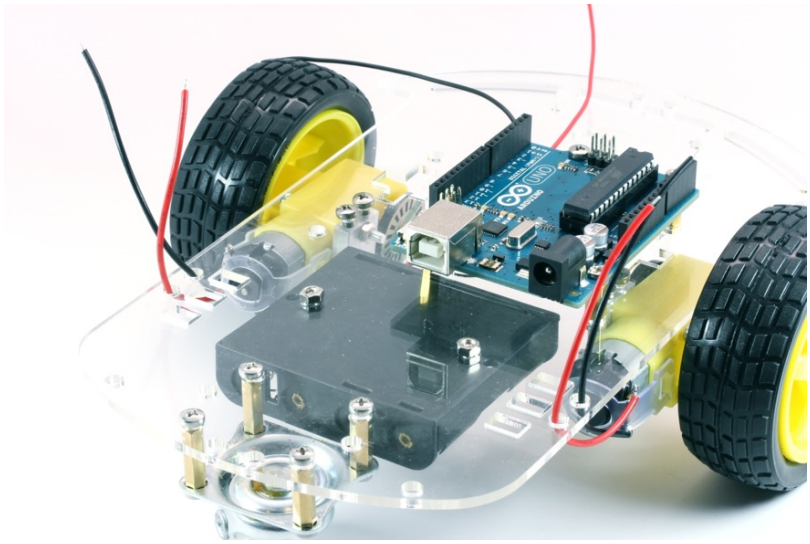
Alternative positions for battery cases and breadboard – use two-sided tape.



Battery box installed underneath.

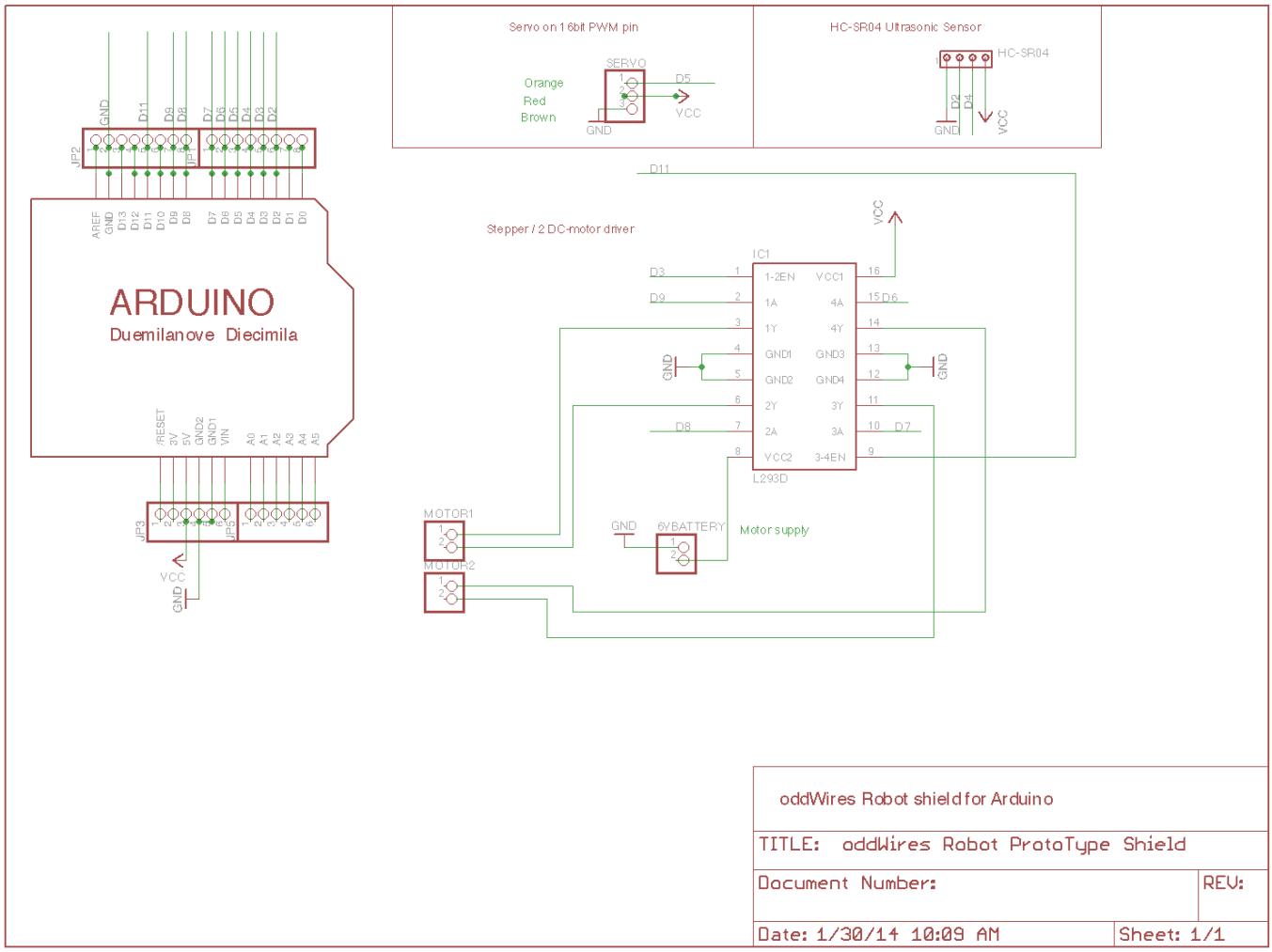


Another view of the battery box installation.



# Constructing the Prototype Board

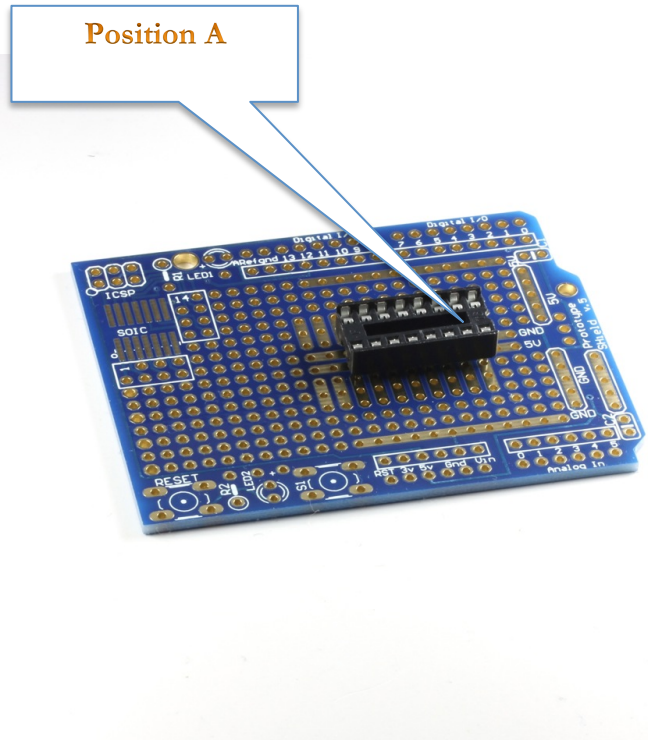
Use the supplied connecting wire to connect the L293D to the Arduino pins. Here's the schematic for the prototype board:



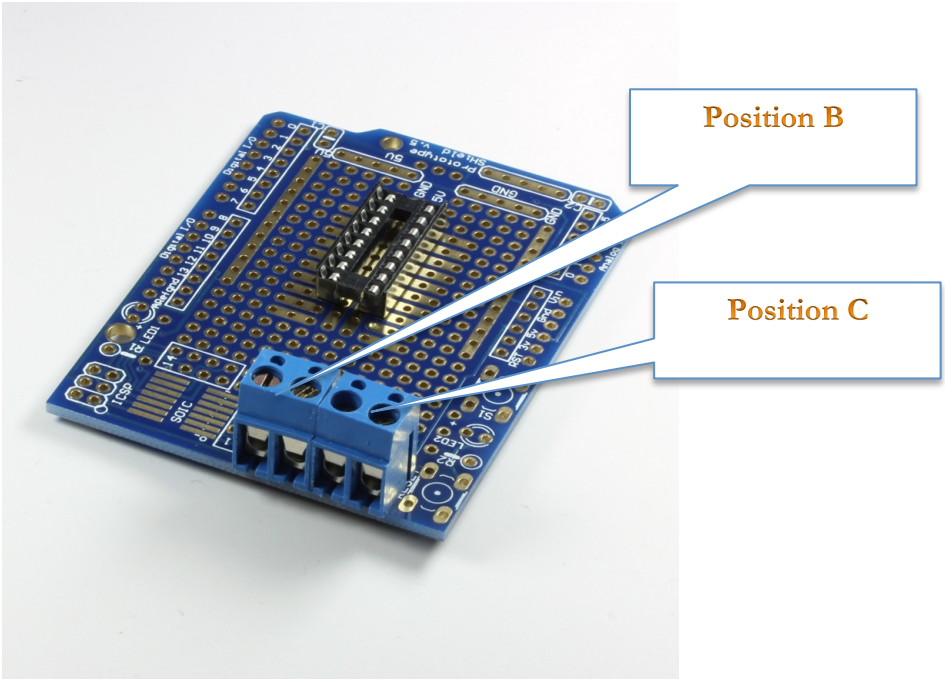

You are going to create the connections between the L293D and the Arduino that are represented in the schematic above. You are also going to use any unused 3-pin position to place the servo connector and wire it up as per the schematic. The Ultrasonic sensor can be wired directly to the Arduino prototype board using the supplied 4 pin M-F connector.

# Wiring the L293D Motor Controller

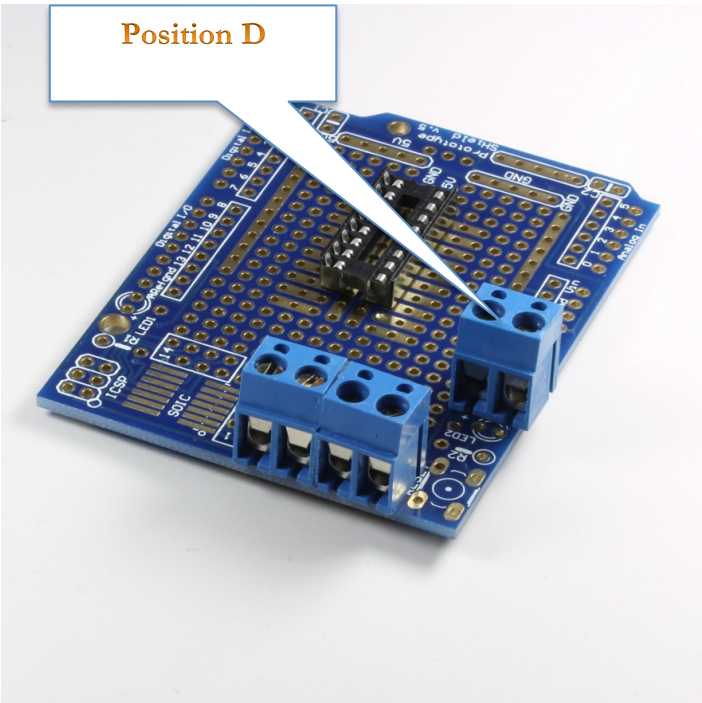
1 Solder 16-pin socket at position A




2 Solder motor terminals (Blue 2-position connector) to board at position B &C



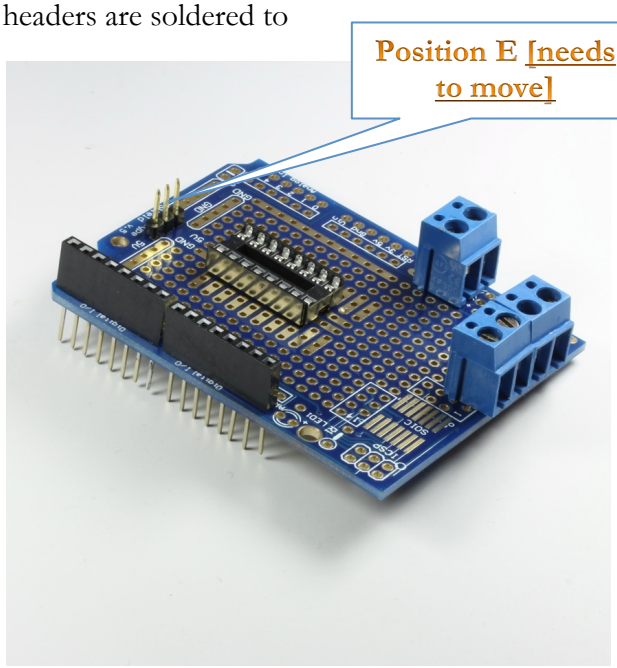
3 Solder battery case input terminal at position D, connect + to Pin 8 and - to GND Rail




| | | |
|---|---|---|
| 4 | GND | Connect Pins 4, 5, 12, 13 to GND Rail |
| 5 | Power | Connect Pin 16 to +5V Rail |
| 6 | Motor 1 | Connect Pin 1 to Arduino Pin 3 (Digital I/O)<br>Connect Pin 2 to Arduino Pin 9 (Digital I/O)<br>Connect Pin 7 to Arduino Pin 8 (Digital I/O)<br>Connect Pin 3 & 6 to Motor 1 terminal |
| 7 | Motor 2 | Connect Pin 9 to Arduino Pin 11 (Digital I/O)<br>Connect Pin 10 to Arduino Pin 7 (Digital I/O)<br>Connect Pin 15 to Arduino Pin 6 (Digital I/O)<br>Connect Pin 11 & 14 to Motor 2 terminal |

8 Solder stackable headers 2 x 6 pin
Solder stackable headers 2 x 8 pin

Two 8-Pin stackable headers soldered to board. Note that they are to the outside of the board. There are pins side of these that you solder the connecting wires between the headers and the L293D chip. The 6-Pin headers are soldered to the other side of the prototype board.

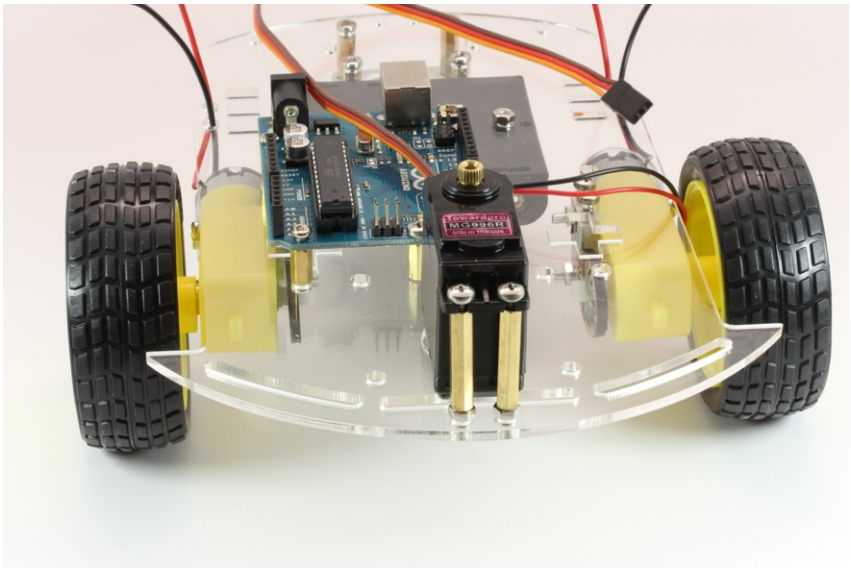



**Overall Summary of L293D Connections**

| Pin | Function | Connection | |
|---|---|---|---|
| 1 | ENABLE1 | Arduino Pin 3 | |
| 2 | INPUT1 | Arduino Pin 9 | |
| 3 | OUTPUT1 | Motor 1 | |
| 4 | GND | Arduino GND | |
| 5 | GND | Arduino GND | |
| 6 | OUTPUT2 | Motor 1 | |
| 7 | INPUT2 | Arduino Pin 8 | |
| 8 | Vs | Motor drive supply | Connect to 6V Battery Case **NOT** Arduino +5V |
| 9 | ENABLE2 | Arduino Pin 11 | |
| 10 | INPUT3 | Arduino Pin 7 | |
| 11 | OUTPUT3 | Motor 2 | |
| 12 | GND | Arduino GND | |
| 13 | GND | Arduino GND | |
| 14 | OUTPUT4 | Motor 2 | |
| 15 | INPUT4 | Arduino Pin 6 | |
| 16 | Vss | Arduino +5V | |

# Installing the Servo

Servo attachment. Note the nut below the standoff as a spacer.



Solder 3-pin male header of the 40-pin connector to Position E

Solder the pins as follows

| | |
|---|---|
| Brown | GND |
| Red | Motor drive supply (6V battery box in this case - do not use Arduino +5V) |
| Orange | Arduino Pin 5 |

# Installing the Ultrasonic Distance Sensor

Use the 4-pin male-female connector to connect the Ultrasonic sensor as follows:

| **Sensor** | **Arduino** |
|---|---|
| Vcc | Arduino +5V |
| Trig | Arduino Pin 4 |
| Echo | Arduino Pin 2 |
| GND | Arduino GND |

The ultrasonic distance sensor is attached to the servo mount using double-sided tape. There is little stress on this item, but if wish you can attach it using contact cement.

# Final Construction and Testing

Connect servo to 3-pin connector on prototype board. Wire up motors to motor terminals. Wire battery box to battery input terminals. Use white connectors to join wires where necessary. 0.1uF caps across motors if noisy.

# Ideas for Extending your Robot

Here are a few ideas for extending your Arduino-based robot.

## LDR (moth)

Use a Light Dependent Resistor as part of a voltage divider to sense the light. Read the values from an Arduino analog pin and move the robot to the source of light.

## PIR Sensor (movement sensor)

Use a PIR sensor to detect movement. Have it chase a person moving around the room.

## IR Remote Control

Use a TSOP3848 IR receiver in conjunction with a remote control (or build your own with a TSAL7400 IR LED and an Arduino) to remotely control your robot.