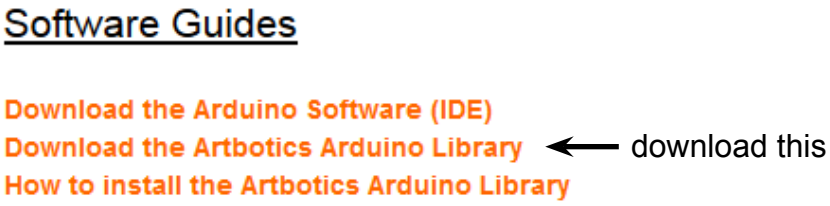
Installing the Artbotics library

Note: You must install and run the Arduino IDE at least once before installing Artbotics. Please go to arduino.cc/en/Main/Software and follow the installation instructions for your system and run the program before continuing.

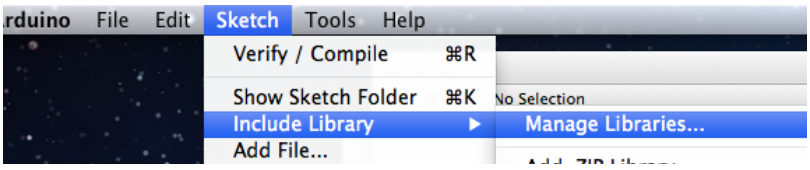
Step 1 Download the Artbotics library from our website Artbotics.com under **How to**, **Arduino,** and **Arduino C++.** The link to the library will be under **Software Guides** shown below.



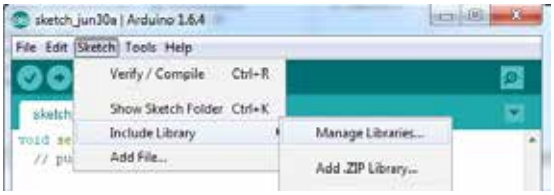
Step 2 Once downloaded unzip the contents and copy the **Artbotics** folder into your **Documents/ Arduino/libraries** folder.

Note: The **Arduino** and **libraries** folders are generated the first time you run the Arduino IDE.

Step 3 Now open up your Arduino IDE and go to **Sketch**, **Include Library**, then **Manage Libraries**



Mac OSX



Windows

Let the library manager load for a moment and when it is finished, close the **library manager window**. Now when you go back to **Sketch** and **Include library,** at the bottom of the drop down list you should see **Artbotics**. Double click **Artbotics** to add it to your sketch (code).

This should appear at the bottom of the list

Objects

Artbotics has four main object:

Car used to control both motors together for drive and draw. (below)

Motor used to control a single motor for a mechanical sculpture (page 3).

Sensor analog sensor used to signal a reaction for a mechanical sculpture (page 4).

Button binary touch sensor used to signal a reaction for a mechanical sculpture (page 4).

Set up and using the Car object

The Car object is designed to run both motors easily as if you were driving a car, so when you instantiate it, you must pass the pins for both motors. We recommend you use the labels on the motor shield to identify the pins for motor terminal A and motor terminal B.

Instantiation Example:
Car car(3, 12, 11, 13); // instantiates a car object to control motor A and B on a standard motor shield

Car has two functions to move the car forward and backward take two arguments. The first is duration which is a number in seconds that will have the car move forward or backward for that amount of time. The second argument is power which is a value between 0(off) and 5(full power) which determines how fast the car will move.

Examples of move forward and backward:
car.moveForward(duration, power);
car.moveBackward(duration, power);

The next two functions are turn in place functions, the arguments are the same.

Examples of turn left and right (in place):
car.turnLeft(duration, power);
car.turnRight(duration, power);

The next two functions are arc turns, the first two arguments are the same, the arc argument is a integer from 0 (straight) and 5 (in place turn)
car.arcLeft(duration, power, arc);
car.arcRight(duration, power, arc);

The last function will have the car pivot in place, with a normal duration and power argument. dir is a boolean value of either HIGH or LOW that determines which direction the car will spin. Rotation is determined by the wiring of the motors so testing is required to figure out which direction will give you the turn you want.

car.spin(duration, power, dir);

Set up and using the Motor object

The Motor object is designed to run a single motor, so when you instantiate it, you must pass the pins for a single motor (either motor A or motor B).

Instantiation Examples:
Motor motorA(3, 12); // will instantiate motorA to control the motor A terminal.
Motor motorB(11, 13); // will instantiate motorB to control the motor B terminal.

Motor's function:

Motor has two functions, the main function you would use is **rotate** while passing it three arguments.

Three argument rotate example:
motorA.rotate(duration, power, direction); // the first two arguments are numbers, the last is boolean

This version of rotate will rotate the motor for the **duration** value, passed in seconds. With a **power** value between 0(off) and 5(full power). Also a given **direction** which is either HIGH or LOW (must be in all caps).

Rotate also has a two argument version which **must be used with stop()**. Stop is used to turn off motors which were turned on with two argument rotate.

Two argument rotate example:
motorA.rotate(power, direction); // the first argument is a number, the last is boolean

This version of rotate will rotate a motor until turned off with **stop()**. The power and direction arguments operate the same as before.

Examples using rotate and stop:
motorA.rotate(2, 3, HIGH); // will rotate motorA for 2 seconds at power 3 in the high direction
motorA.rotate(3, HIGH); // will continuously rotate motorA at power 3 in the high direction
motorA.stop(); // will stop motorA following a motorA.rotate(power, direction);

Set up and using the Sensor/Button object

A sensor will read a given pin and assuming it has a sensor attached to that pin will give you a value based on what the sensor is reading at that moment. The value will fall between 0 and 1023 though the actual range of a sensor will vary.

Another variable of a sensor is whether 0 or 1023 represents a low sensor reading like low lighting or a low sound level. To test the sensor you must create a Serial object and print the values of the sensor readings to it, while getting readings expose the sensor to what would evaluate to a low or high reading. For example if you are using a light sensor cover it with your hand, or for a sound sensor clap your hands close to it. This will give you a value spike, either closer to 0 or 1023, use this information to program a reaction in a simple exercise.

Instantiation Example:
Sensor lightSensor(2); // will instantiate a sensor to read analog pin 2.
Sensor soundSensor(3); // will instantiate a sensor to read analog pin 3.

note: analog pins 2 through 5 can be used for sensors but while using the Arduino motor shield provided by Arduino, pins 0 and 1 can't be used

The main function used with sensor is **getReading**. This function will return a value between 0 and 1023 which you would use to trigger a reaction.

Function Example:
lightSensor.getReading(); // returns an integer representing the sensor reading from analog pin 2
soundSensor.getReading(); //returns an integer representing the sensor reading form analog pin 3

A button is a touch sensor which will only return a boolean value (HIGH or LOW). HIGH typically meaning the button is pressed and LOW typically meaning the button is not pressed.

Instantiation Example:
Button button_2(2); // will instantiate a button to read digital pin 2.

Note: buttons will use a digital pin not an analog pin.