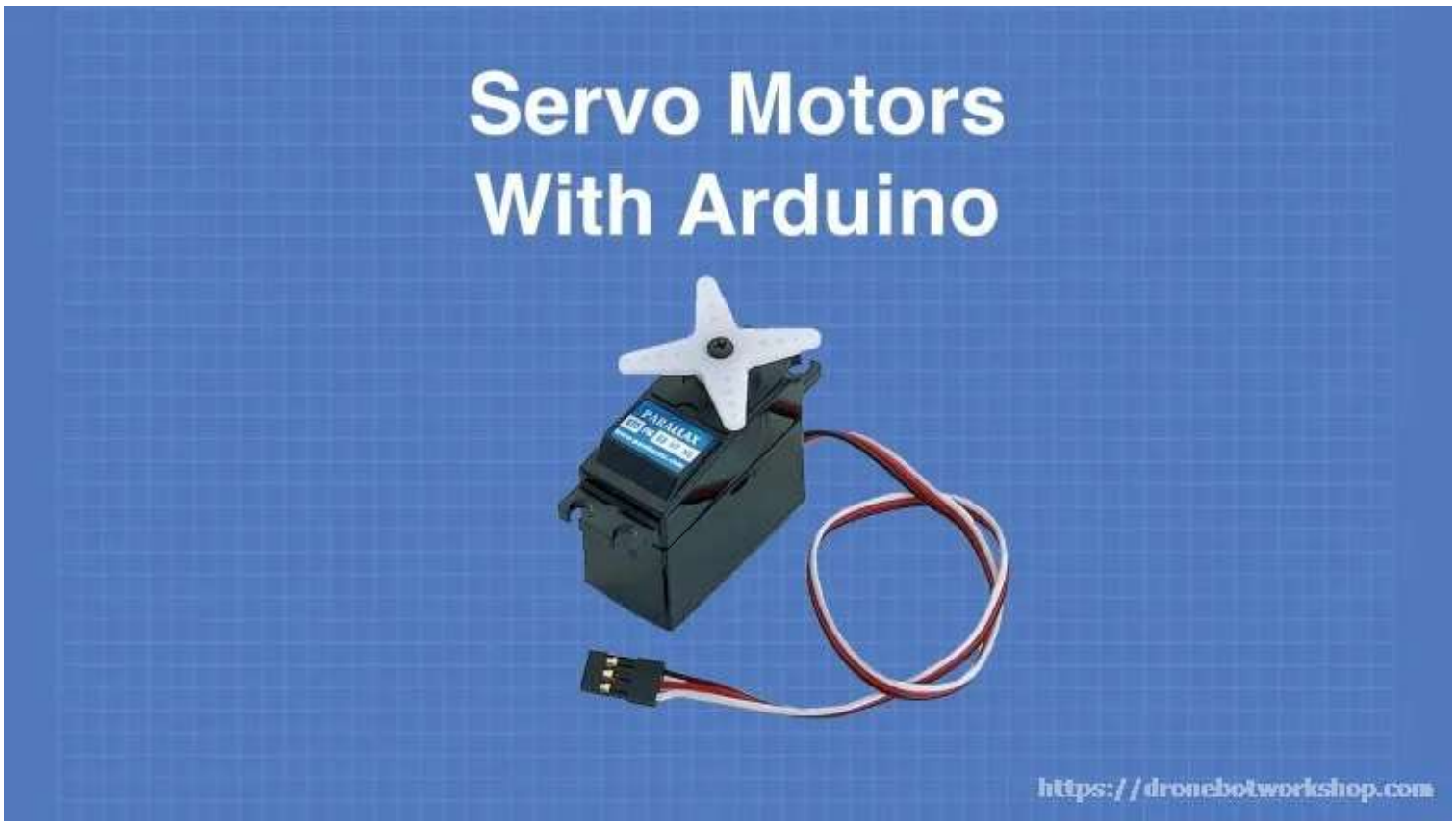
What is a Servo Motor?

We have built a few projects that have used motors to make things move and along the way we have looked at some of the different types of motors that we can control with our Arduino and Raspberry Pi projects.

We have worked with basic DC motors a few times. We built a couple of robotics projects that are based upon DC motors and we also took an extensive look at the H-Bridge Controller that is commonly used to regulate the speed and direction of a DC motor with a microcontroller or microcomputer.

Another type of motor we've worked with is the stepper motor. This type of motor has its shaft driven in discrete steps, allowing for very precise control. They are widely used in printer and robotics designs.

There is another type of motor that we have used in many of our experiments but have not (yet) taken a detailed look at – the Servo Motor.



A Servo Motor is a low-speed, high-torque motor that comes in a variety of sizes. Unlike the DC and Stepper motors the Servo Motor does not normally spin a full 360 degree rotation. Instead it is limited to a range of 180, 270 or 90 degrees.

A control signal is sent to the servo to position the shaft at the desired angle. This arrangement with a single signal makes it simple fo servos to be used in radio and remote controlled designs, as well as with microcontrollers.

A servo is perfect if you need to position the rudder on a boat or the elevator on an aeroplane. They are really useful in robotic work to position cameras, sensors or robot appendages.

Servos can also be used as analog gauges like speedometers and tachometers.

Types of Servo Motors

A servo motor is essentially a motor that has an input for a control signal that is used to specify the position of the motor shaft.

Servos are used in industry as well as in hobby applications. Industrial servos are often AC motors with digital control inputs that cost hundreds or thousands of dollars.

We will NOT be working with industrial servo motors today!

Hobbyist servo motors are generally DC motors that can be controlled with either a digital or analog signal.

Digital servos are used in applications that require quick responses like the elevator on an aeroplane or the rudder on a helicopter. We will NOT be working with these types of motors either, although the hookup and code used to drive them with an Arduino is identical to what we will use for our analog servos.

We will be using plain ordinary analog servo motors, the most popular type for hobbyist use. They are inexpensive and easy to obtain. Mounting hardware is also very easy to find as these servos are of a standard set of sizes.

It should be noted however that while we won't be working with digital servo motors today they are really very much like their analog counterparts. They use the same PWM control signals as analog servo motors and can be controlled using the same circuitry and code.

Analog Servo Motors

Analog servo motors are inexpensive and available in a variety of sizes and ratings. Perfect when you need a tiny high-torque motor that can be accurately positioned and that won't break the bank.

The "analog" part of the analog servo motor is the control signal. Analog servo motors respond to a Pulse Width Modulation or PWM signal to position their motor shaft.

PWM is an ideal control medium. It can be generated by a simple timer circuit or with a microcontroller. It can be sent over a single wire or transmitted on a radio or light beam.

The Arduino has a number of PWM capable output pins, making it ideal for controlling servo motors.

How do Servos Work?

A servo motor is a motor with a built-in "servomechanism".

The servomechanism uses a sensor to monitor the motor shaft position and a controller to control the motor. It is fed a signal that indicates the position that the shaft should be set to. It then moves the motor into the required position.

In the analog servo motors we will be working with that control signal is a PWM signal whose pulse width determines the angle the motor shaft is to be positioned at. The motor itself is a simple DC motor with a lot of gearing to slow down its speed and to increase its torque.

In order to function properly the servo motor needs a sensor that can accurately measure its shaft position. On some industrial and high-end hobby servos this is done using an optical interrupter disc, but in most standard hobby servo motors the sensor is a potentiometer. This works well as these servos typically travel 180 to 270 degrees, well within the range of a potentiometer. However the accuracy of potentiometers, especially in low cost servo motors, can affect the overall accuracy of the servomechanism.

Continuous Rotation Servo Motors

A standard analog servo motor is constricted in its rotation, usually to 180 or 270 degrees (180 is by far the most common). Its internal gearing provides a high torque power pack in a small and inexpensive package.

That combination of small size and large torque also make servos attractive to use as replacements for standard DC motors in the design of small devices like tiny toys and robots. This prompted several people to modify standard analog servos by removing the potentiometer to allow the servo to spin a full 360 degrees.

Manufactures got the message and now offer "continuous rotation servo motors", essentially servos with the servomechanism disengaged.

In a continuous rotation servo motor the speed and direction of the shaft rotation is controlled by the same PWM signal that is used in a conventional analog servo motor.

Having a simple one-wire controls signal and the same physical package as a standard servo motor make continuous rotation servo motors attractive for a number of applications.

Servo Motor Control Signals

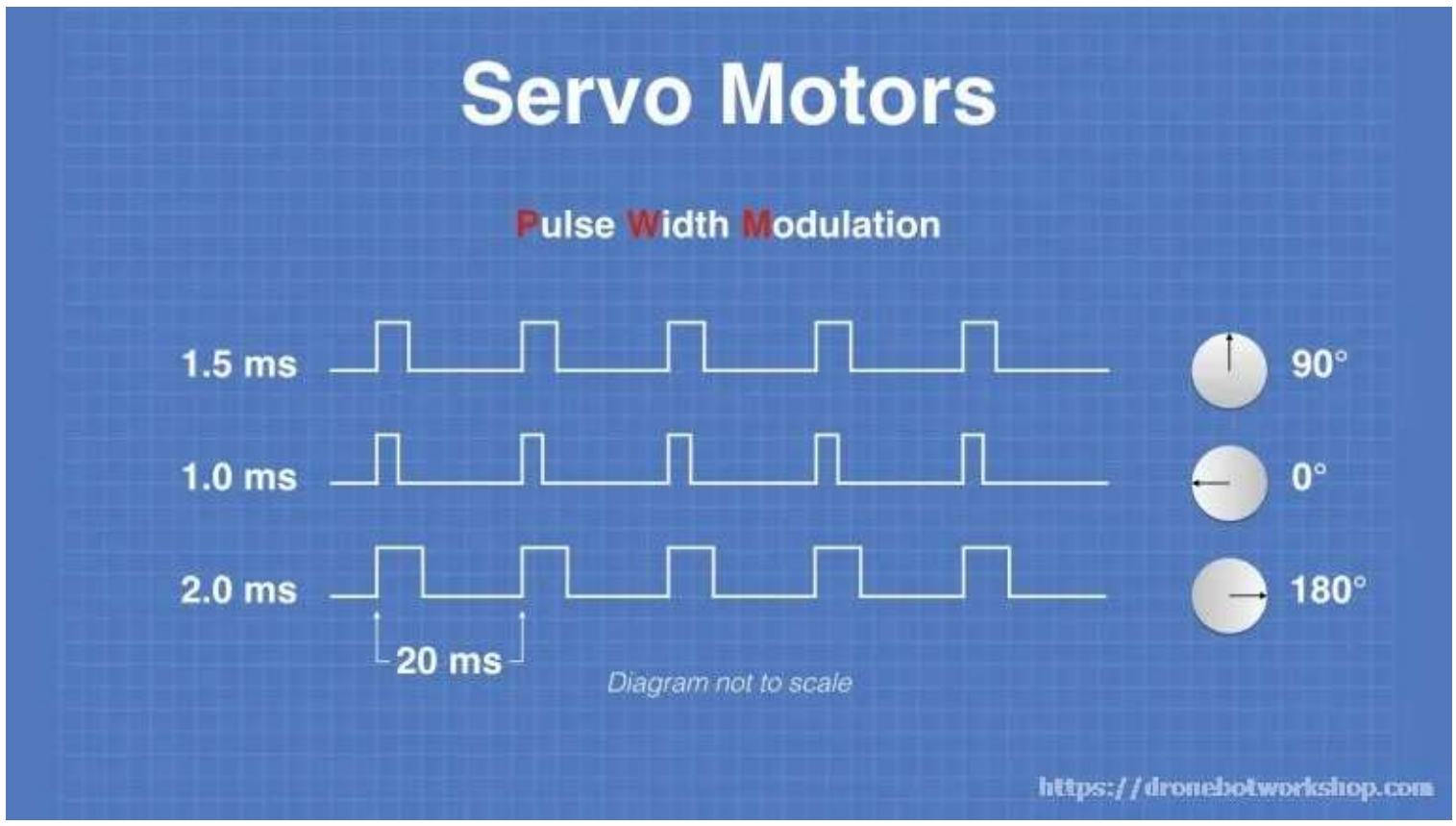
In order to use analog servo motors you need to understand how to control their operation using PWM. The two varieties, conventional and continuous rotation, use the same timing signals but respond to them slightly differently.

Let's take a look at the PWM signals used in most analog servo motors.

Conventional Servo Motor Timing

In a conventional analog servo motor a PWM signal with a period of 20 ms is used to control the motors. A signal of 20 ms has a frequency of 50 Hz.

The width of the pulse is varied between 1 and 2 ms to control the motor shaft position.

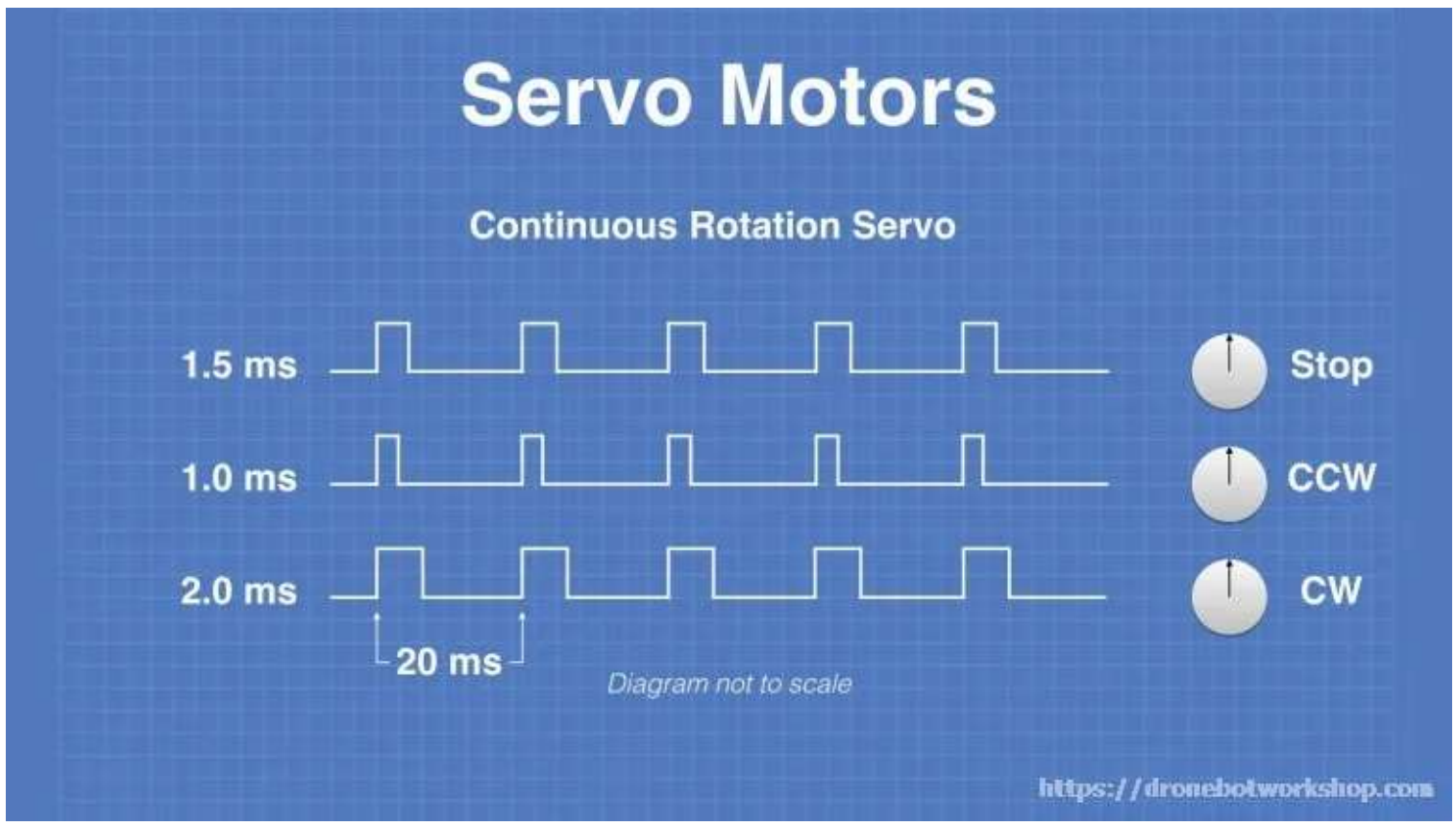


- A pulse width of 1.5ms will cause the servo shaft to rest in the 90 degree position, the center of its travel.
- A pulse width of 1ms will cause the servo shaft to rest at the 0 degree position.
- A pulse width of 2ms will cause the servo shaft to rest in the 180 degree position.

Varying the pulse width between 1ms and 2ms will move the servo shaft through the full 180 degrees of its travel. You can bring it to rest at any angle you desire by adjusting the pulse width accordingly.

Continuous Rotation Servo Motor Timing

In a continuous rotation servo motor the same PWM signals will cause the motor to perform differently.



- A pulse width of 1.5ms will cause the servo shaft stop spinning.
- A pulse width of 1ms will cause the servo shaft to spin at full speed counter-clockwise..
- A pulse width of 2ms will cause the servo shaft to spin at full speed clockwise.

Varying the pulse width between 1ms and 1.5ms will make the motor spin counterclockwise with the shorter pulse widths causing the motor to spin faster.

Varying the pulse width between 1.5ms and 2ms will cause the motor to rotate clockwise with the longer pulses resulting in a faster speed.

Commercial continuous rotation servo motors will have an adjustment potentiometer that can be used to zero the speed when the motor is feed a 1.5ms pulse width.

Servo Motor Specifications

There are literally hundreds of analog servo motors available, knowing how to read their specifications is essential to choosing the correct one for your application.

Here are some of the key parameters you'll encounter when choosing a servo motor.

Motor Size

The physical size of a servo motor is naturally an important consideration, chances are your application will demand that the motor conform to specific size restrictions.

There are a number of standard servo motor sizes, this makes it a lot easier to find mounting brackets and hardware to accommodate your servo motor.

Servo sizes are often specified as follows:

- Nano
- Sub-Micro
- Micro
- Mini
- Standard
- Large

There are also specialty sizes. The Micro and Standard sizes are the ones most often used by Arduino experimenters.

Gear Material

Servos have a number of internal gears plus gearing directly on the output shaft, these serve to slow down the motor speed and increase its torque.

The gears can be manufactured using either plastic or metal.

Metal gears offer better performance, can usually support higher torques and are less subject toi stripping. Metal gear servos also come at a higher cost.

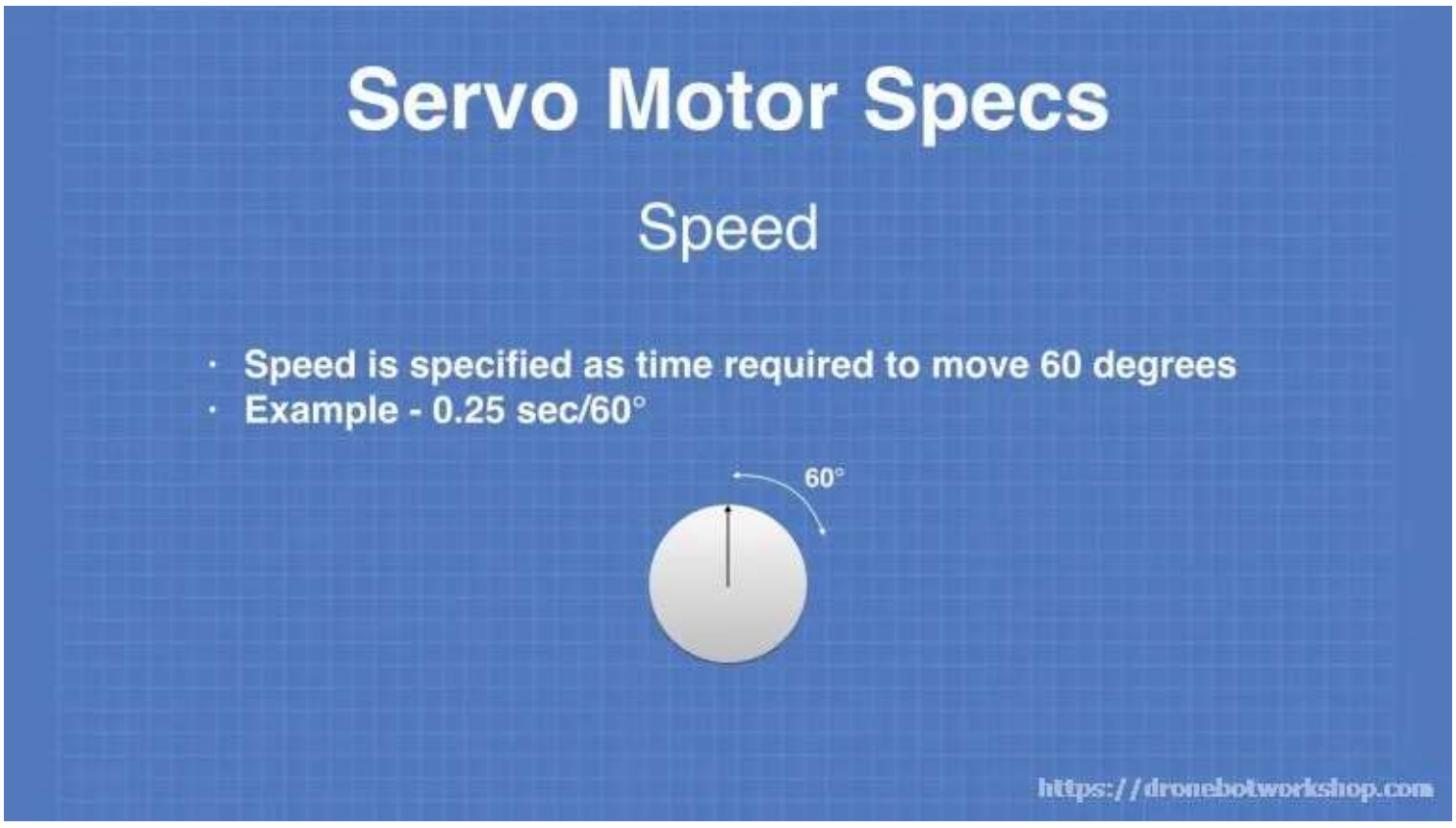
Plastic gears are more susceptible to stripping and don't have the torque capabilities of their metal counterparts. They are however quieter and are less expensive than metal geared servo motors.

You can often get the same servo motor with a choice of gears. A common experimenters motor is the SG90, a Micro sized servo motor with plastic gears. Its metal-geared counterpart is the MG90. As they come in the same case and have the same voltage and driver requirements they are interchangeable, with the MG90 offering superior performance because of its metal gears.

Servo motor quality is also affected by the type of bearings and the number of them. Motors with multiple bearings have smoother and more accurate rotation.

Speed

The speed of a servo motor is specified as the time required to move the shaft of the servo by 60 degrees.



An example rating is 0.25 sec/60°, meaning it takes a quarter of a second to travel 60 degrees.

Servo speed is more applicable to conventional servo motors, continuous rotation motors are rated in maximum RPM like any DC motor.

High speed servos are used in model aeroplane and helicopter application to control elevators and rudders which often need to be moved quickly. Many of these use digital control and internal optical position sensors instead of potentiometers to allow for more rapid movement.

For hobby applications like controlling the position of a video camera or ultrasonic sensor speed isn't usually a critical factor.

Torque

Torque is a very important parameter, it literally specifies how strong a motor is.

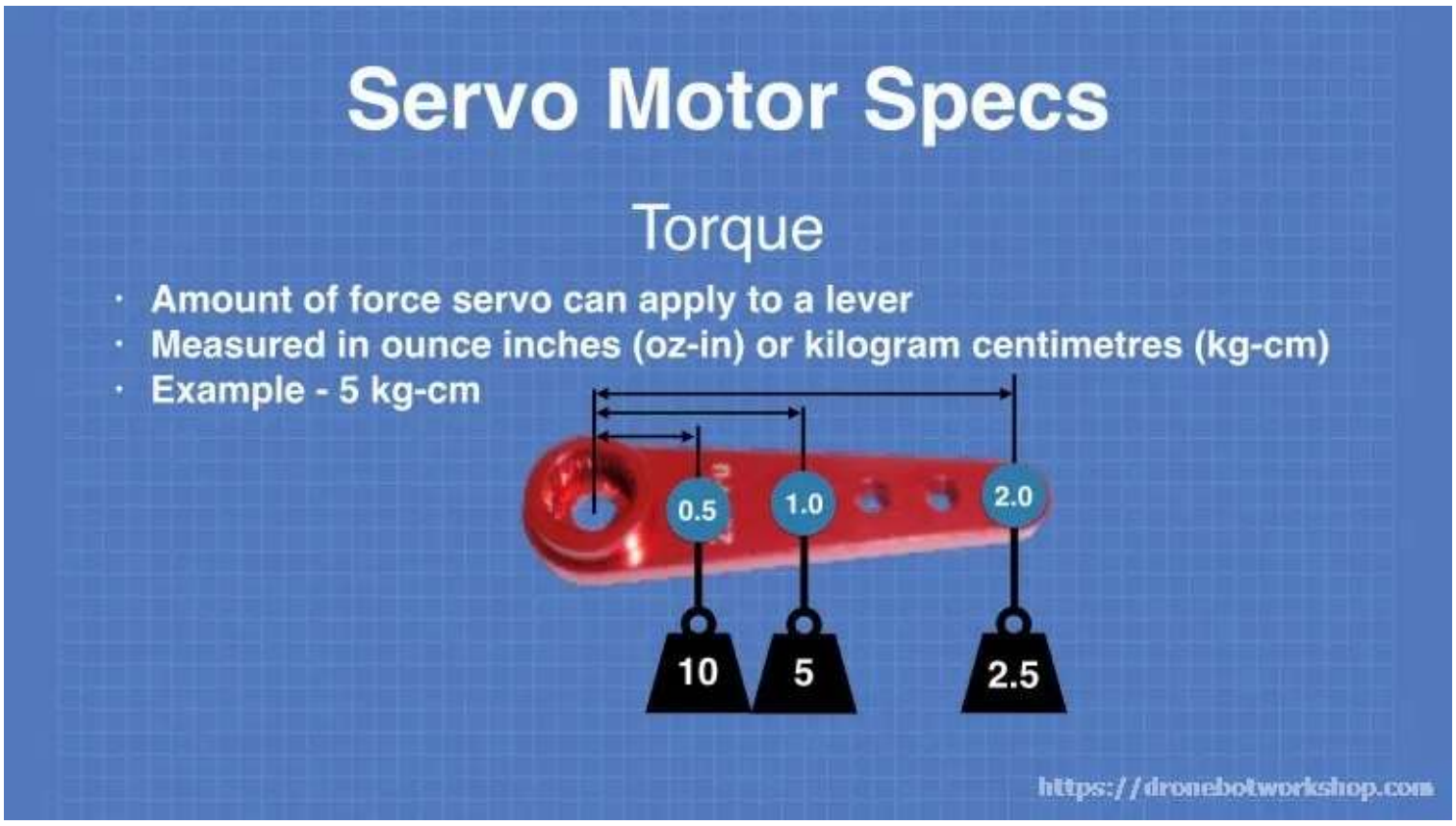
Torque is defined as the amount of force a servo can apply to a lever or in other words how much weight it can hold up.

It is measures in either ounce-inches or kilogram-centimeters. You can convert manually as follows:

- Multiply kilogram-centimeters by 13.88 to get ounce-inches.
- Divide ounce inches by 13.88 to get kilogram-centimeters.

Another way is to use an online torque converter.

To understand how the torque figures relate to real world conditions consider the following example:



A servo motor is rated at 5 kg-cm. This would also be the same as 69.4 oz-in (5 x 13.88 = 69.4).

The servo motor can support up to 5kg of load on a lever at a distance of 1 centimeter from shaft center. If you prefer Imperial measurements then it could support a 69.4 ounce load (about 4.3 pounds) at a distance of 1 inch.

At double the distance the load would be halved, so at 2cm from the shaft the lever could support 2.5kg.

Half the distance doubles the load that can be supported to 10kg.

Larger servo motors tend to have larger torque capabilities, motors with greater torque tend to be more expensive. They also weigh more and consume more current.

Operating Voltage

Most hobby analog servos are rated from 4.8 to 6 volts and achieve their maximum performance at the higher voltage.

There are also more servos being offered with maximum voltage ratings of 7.5 to 8.5 volts. These are becoming more popular due to the availability of 7.4 volt LiPo batteries for model aircraft, boats, vehicles and quadcopters.

Servo motors, especially high torque models, can consume quite a lot of current, this needs to be taken into account when selecting a power supply or battery for your project.

Horns, Arms and Accessories

Most servo motors have a geared shaft that is threaded for a center screw.

In order to make use of the servo you will need to connect the shift to another component in your design – a platform, a gear, a wheel or whatever it is you are trying to move with the servo.

Servo motors come with an assortment of levers and discs of different shapes that can be mated to the shaft to facilitate attaching the servo to your design. These pieces are often referred to as “horns” or “arms”. They attach to the servo motor shaft and are secured in position with the center screw and they can be made of plastic or metal.

In addition to the horns and arms you should also receive an assortment of mounting hardware and screws with your servo, including the center screw for the shaft (don’t lose it as they tend to differ between servo types).

You can also buy mounting plates designed to accept popular servo sizes like Micro and Standard.

The availability and interchangeability of servo horns, arms, mounting hardware and accessories makes it easy to incorporate servo motors into your designs.

Testing Servo Motors

As with any component its useful to know how to test a servo motor to ensure proper operation.

This can be very helpful when you are about to mount the servo into a mission-critical application, or just into something that would be a pain to have to take apart if the servo turns out to be faulty!

It is also useful to be able to rotate the servo shaft into a preset position (for example 90 degrees) before mounting the sero into your project so that everything gets aligned correctly.

There are a number of methods you can use to test a servo. A simple Arduino sketch and connection like the ones you’ll be seeing here further on will make an excellent method of testing a servo and of positioning its shaft into a preset position.

Another way of doing this is by using a dedicated servo tester.

Servo Testers

As you might imagine a Servo Tester is a device used to test servo motors! They are very useful and can be very inexpensive, depending upon the features you want.

A simple servo tester like the one shown here (and used in the accompanying video) can be had for just a couple of dollars.

More advance servo testers have speed and centering controls for multiple motors, some also have current meters. Even these are under 20 dollars.

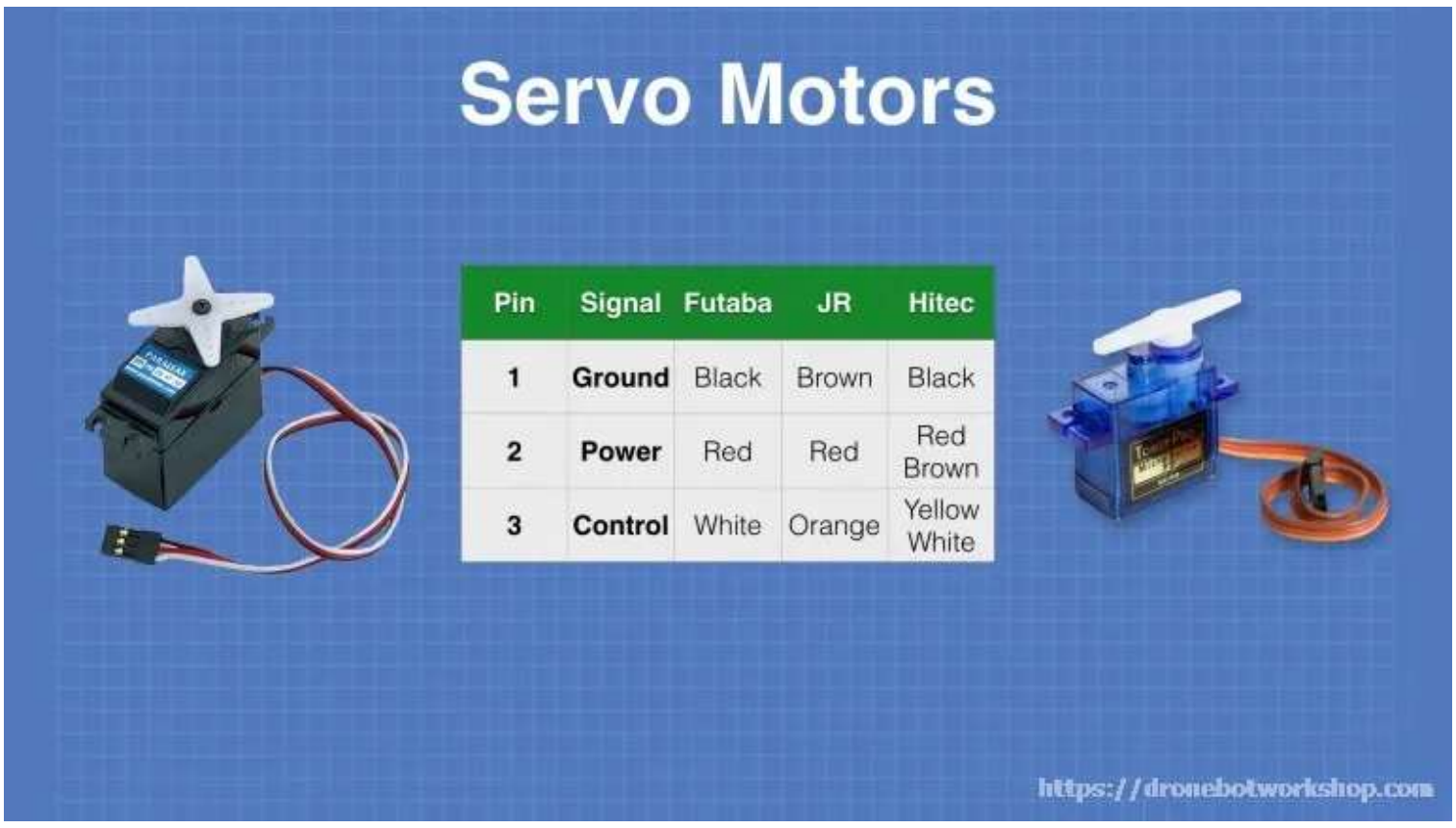
These units need to be powered by the same power supply (or equivalent) that you’ll be using the power the motors themselves. They can be plugged in to a standard servo motor connector and they will take control of the servo.

The servo testers will allow you to manually move the motor and to center it at the 90 degree position. This lets you check the motor for correct operation and to align its shaft position before you fasten horns or arms to it.

Servo Motor Connections

Analog servo motors typically have a 3-pin connector. On some more expensive motors the motor cable can be removed at the motor base and replaced if required, other motors have the connector permanently wired onto a short 3-wire cable.

The color codes used on hobbyist servo motors varies depending upon manufacturer. However most manufacturers observe the same pinout, as shown in the following diagram:



The three connections to the servo motors are as follows:

1. **Ground** – The common ground for both the motor and logic.
2. **Power** – The positive voltage that powers the servo.
3. **Control** – The input for the PWM control signal.

The most common connector is the standard DuPont variety with 0.1 inch spacing. This makes it easy to connect servo motors to your project using standard DuPont header strips.

You can also insert breadboard wires directly into the servos 3-pin connector so you can prototype with servo motors.

Connecting to an Arduino

As we have already described a servo motor requires a PWM control signal to operate correctly. You can generate this signal many ways – a simple timer circuit, a dedicated control chip or using a microcontroller with PWM output capabilities.

The microcontroller naturally has many advantages in being able to control the servo more effectively. And an Arduino is an excellent choice.

The Arduino IDE has a Servo library already included so adding a servo to your sketch is very simple as you will soon see.

PWM Output

All Arduino boards have some output pins that are capable of Pulse Width Modulation or PWM. On an Arduino Uno there are 6 PWM-capable pins.

Keep in mind that to generate the PWM signals the Servo Library will need to use some of the internal Arduino timers, specifically Timer 1. This can interfere with other libraries that also need the same timers. One way around this is to look for alternate libraries for either the servo of=or the other desired function, this is a common way of getting around these restrictions.

A more advanced method is to use an external PWM controller board and free up the Arduino timers. This will be discussed further in this article.

Power Supply Considerations

Most servo motors will happily operate on 5 volts, making it tempting to use the 5-volt output on the Arduino board to power the servo.

It's not really a very good idea.

Servos can consume a fair amount of current, especially when placed under load. This might be more current than the voltage regulator on the Arduino board can take, especially on cheaper clone boards. While most Arduino boards can support one Micro servo it still taxes the regulator a lot.

Servo motors, like all other motors, can induce electrical noise onto the power supply lines. Having that noise on the lines powering your microcontroller and other logic devices can often lead to system errors.

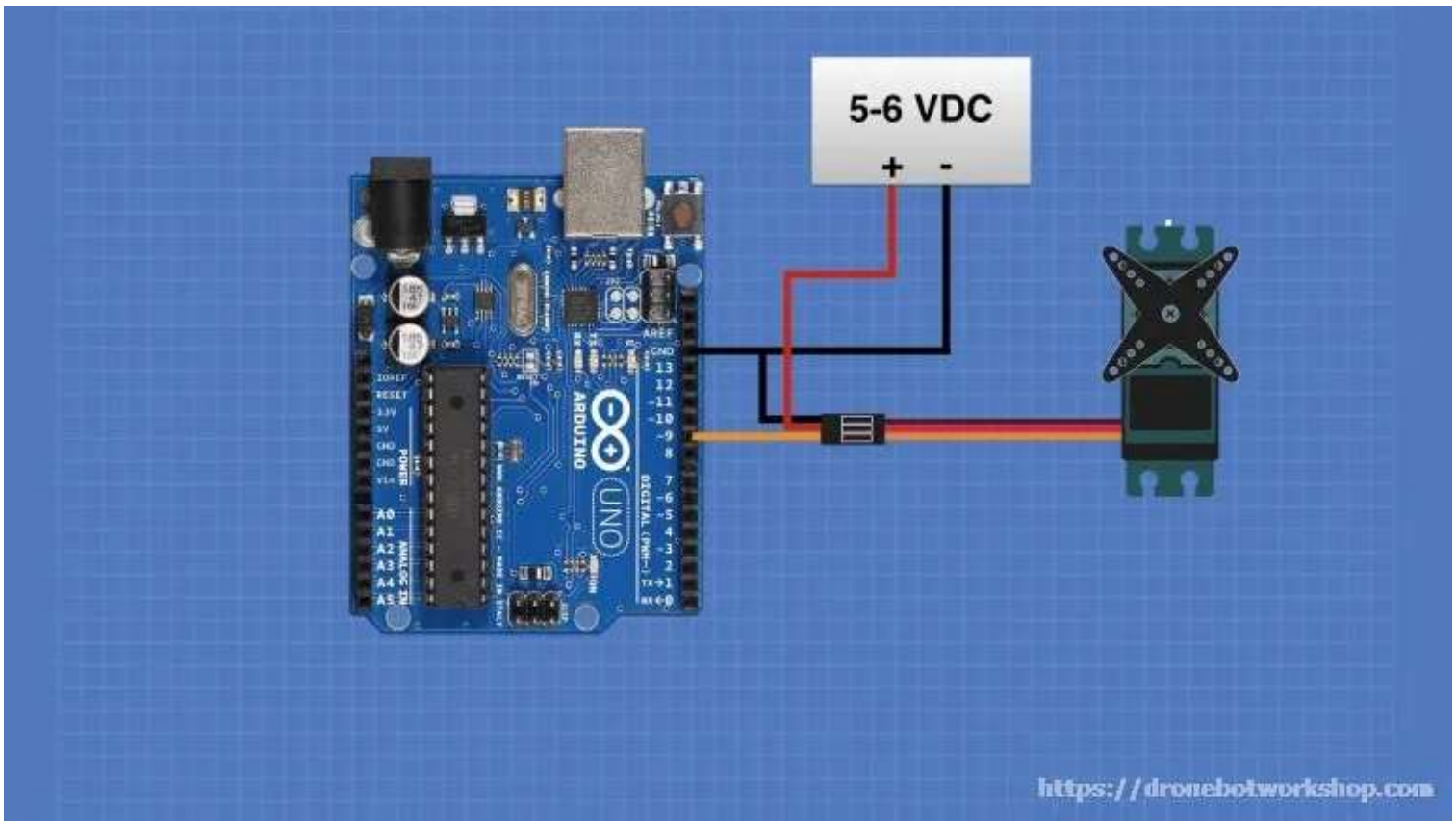
It is a much better idea to use a separate power supply for your servo motor. A 5-volt USB 3 power supply would work well, as would a 6-volt lantern battery or 4 type AA or C batteries.

If you REALLY must power a servo directly from the Arduino limit it to one micro servo. A capacitor of 100uf or greater across the power supply line near the servo can help absorb those power surges.

The Sweep Sketch

For our first Arduino sketch we will use one of the built-in examples that is packaged with your Arduino IDE. No code to write or libraries to install!

Wiring up the hardware to use with our first demonstration is very simple. You'll need an Arduino (any type), a servo motor and a power supply for the servo motor.



The hookup couldn't be simpler. The servo is powered by its own power supply and the ground connection is also connected to the Arduino ground. Then the control lead from the servo is connected to pin 9 on the Arduino.

Pin 9 on the Arduino Uno is one of the six pins that are capable of PWM, on most Uno boards you'll see a symbol beside the 6 PWM-enabled I/O pins.

Hook your Arduino up to your computer and start the Arduino IDE.

Click the *File* menu at the top of the screen.From there select the *Examples* sub-menu.

A list of example sketches will be displayed. It is divided into sections, scroll down the list until you get to the "*Examples from Libraries*" section.

In the "*Examples from Libraries*" section you will see "Servo". Highlight that to reveal two sketches, *Knob* and *Sweep*.

Load the Sweep sketch.

Sweep Sketch

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

 modified 8 Nov 2013
 by Scott Fitzgerald
 http://www.arduino.cc/en/Tutorial/Sweep
*/

#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 positio
  }
  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 positio
  }
}
```

Sweep is a very basic sketch that just sweeps the servo shaft from one extreme to the other.

The sketch makes use of the Arduino Servo Library which is included with your Arduino IDE. As its name implies its is a library for controlling servo motors with PWM. We include the library and define an object called *myservo* to represent our servo motor. If you have multiple servo motors you can define an object for each of them.

We then define a variable called "*pos*" that holds the position (angle) that we want the servo motor shaft to move to.

In the setup we attach our servo object to the servo motor control line on pin 9 of the Arduino.

Then the loop, which consists of two for loops. The first loop increments the value of the *pos* variable and uses it to control the servo motor using a *myservo.write* command, sending the shaft from 0 to 180 degrees.

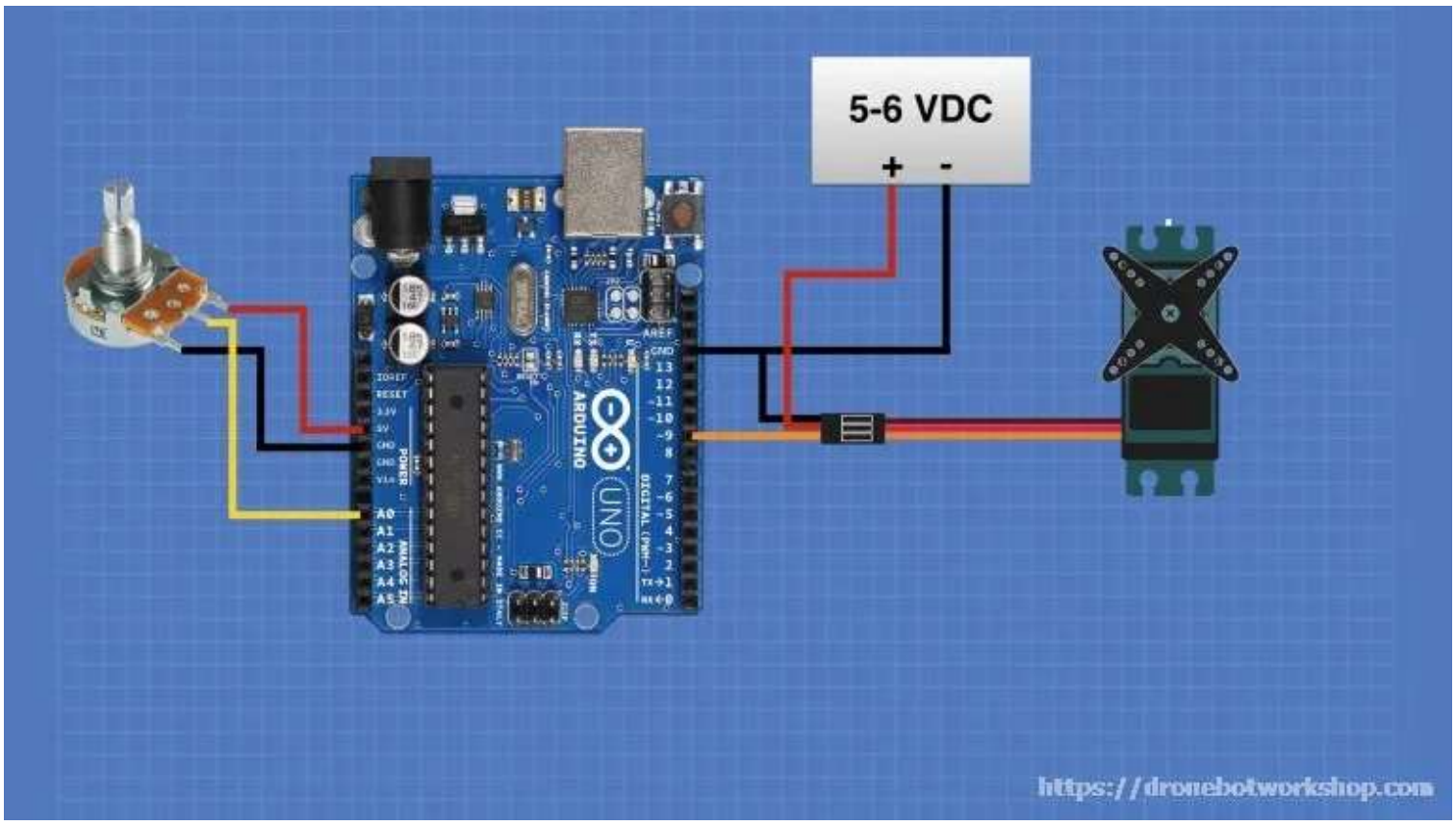
The second for loop is identical except it decrements the value from 180 to 0, sending the shaft back in the opposite direction.

Load the sketch into the Arduino and observe the servo motor shaft, it should be travelling from one end to the other.

You just made a servo move with an Arduino!

The Knob Sketch

Let's move on to the other demo sketch included with the Arduino IDE, the Knob sketch. Before we do we'll need to add a component to our circuit.



As the wiring diagram shows you'll need a potentiometer, any value from 10k up will work fine. Hook one end of the pot to ground, the other end to the Arduino +5 volts. The wiper is connected to analog input A0.

The potentiometer will serve as a control to position the shaft of the servo motor, you can use it to dial any position on its 180 degree travel. Not only is it a good demonstration it also can be a useful function for setting the position of servo motors before mounting them into your project.

If you substitute a continuous rotation servo in the circuit you can use the potentiometer to control both the speed and direction of the motor's rotation.

After you modify the experiment to include the potentiometer open the Arduino IDE and go back to the example sketches. This time select *Sweep* from the *Servo* menu.

Arduino Knob Sketch

```
/*
 Controlling a servo position using a potentiometer (variable resistor)
 by Michal Rinott <http://people.interaction-ivrea.it/m.rinott>

 modified on 8 Nov 2013
 by Scott Fitzgerald
 http://www.arduino.cc/en/Tutorial/Knob
*/

#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
  val = map(val, 0, 1023, 0, 180);   // scale it to use it with the servo (value be
  myservo.write(val);                // sets the servo position according to the sc
  delay(15);                         // waits for the servo to get there
}
```

The Sweep sketch is also very simple. Like the Knob sketch it uses the Arduino Servo Library which it includes and then creates a *myservo* object to represent the servo motor.

We then define a couple of integers. The first one, *potpin*, represents the analog pin we used for the potentiometer wiper connection. The other one, *val*, is the value taken when reading that analog input.

The setup is identical to the Knob sketch, we attach the servo object to pin 9.

In the loop we start by reading the value from the analog pin, a value of 0 to 1023 which will be assigned to *val*. Next we use the Arduino Map Function to change *val* to represent the angle between 0 and 180 degrees.

After that we use a write command to position the servo to the value of *val*, the angle selected by the potentiometer.

After a brief delay to allow the servo motor to catch up we do it all over again.

Load the sketch up to your Arduino and turn the potentiometer. You should see the shaft of the servo motor move in time with the pot.

Once again the sketch illustrates just how easy it is to control a servo with your Arduino.

PCA9685 Servo Driver Board

Controlling servo motors from an Arduino directly is pretty simple as we just saw. However it has its limitations:

- You are limited by the number of PWM pins on your Arduino. If the servo is part of a design that requires other PWM devices that may be a problem.
- The Arduino Servo Library can conflict with other Arduino libraries as they attempt to use the same timer. This can sometimes be solved by looking for alternative libraries.
- You need to control a LOT of servo motors, even an Arduino Mega has its limitations here.

A better solution all around is to use a separate servo driver board. This will offload the task of sending PWM to the servos, freeing up your Arduino to do better things.

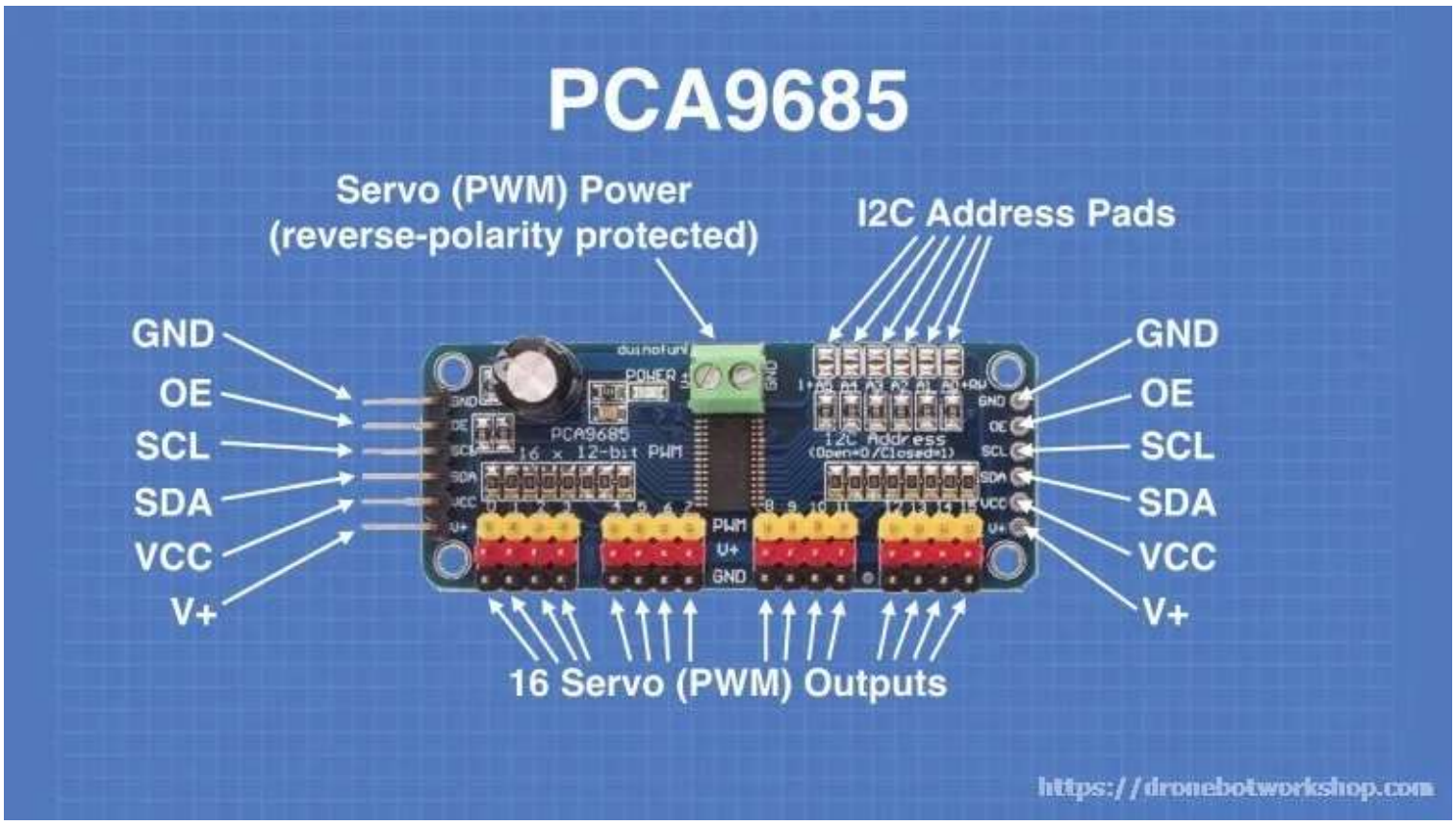
The board we will be using is based around the PCA9685 chip. These boards are extremely popular and are manufactured by several companies.

The PCA9685 board uses I2C to communicate with the Arduino. This means only two connections for clock and data are made to the Arduino. As the boards I2C address can be configured using a series of solder pads you can use many of them on the same circuit.

Each board can control up to 16 servo motors. And you can cascade up to 62 boards to control a whopping 992 servo motors!

If you honestly need to control over 992 servo motors you could use an I2C shield to connect multiple I2C buses to your Arduino!

The connections to the board are very simple.



There are a set of identical connections on each side of the circuit board, this makes it easy to connect several modules up in a row. They are as follows:

- **GND** – The Ground connection.
- **OE** – Output enable. You can use this pin to enable and disable all of the 16 outputs. Typically it is left unconnected which will result in all outputs being enabled.
- **SCL** – The Clock line for the I2C bus.
- **SDA** – The Data line for the I2C bus.
- **VCC** – The logic power supply, +5 Volts.
- **V+** – the power for the servo motors. There is also another connector on top of the board for this and that connector is preferable as it is reverse polarity protected while V+ is not. The V+ pins are really used to cascade multiple PCA9685 modules and power all the servos off of a single power supply.

There is also a 2-pin screw connector at the top for the servo power supply. As mentioned above it is reverse polarity protected.

On the bottom of the board are 16 sets of 3-pin male connectors. Each one is used for a servo motor.

On the top right of the board are six solder pads. These are used to setup the I2C address for the board. If you are using more than one board you'll need to jumper one or more of these to change its internal I2C address to be unique.

The base address for a PCA9685 module with none of the jumpers shorted is 0x40.

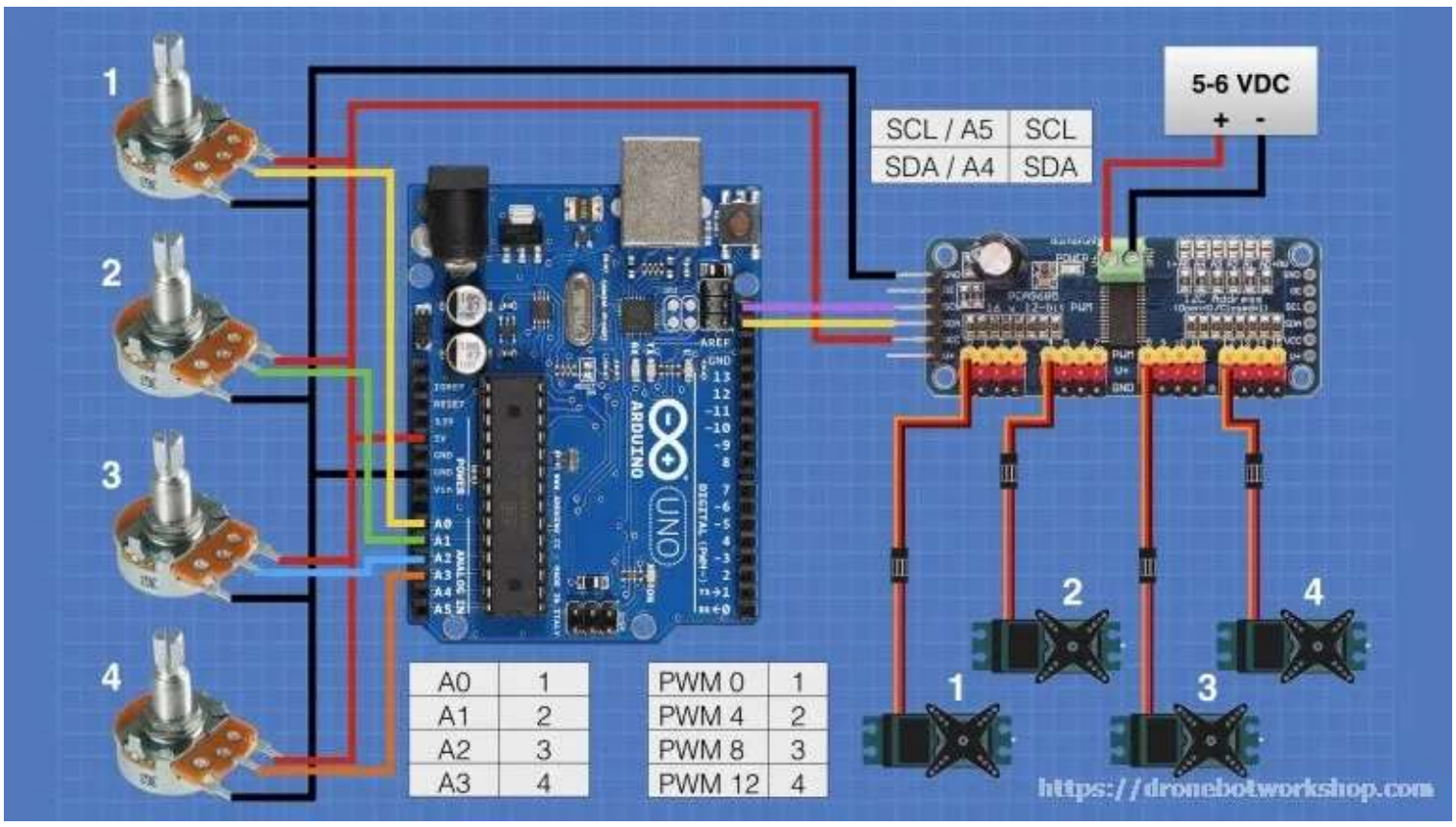
If you short out the A0 solder pad the address becomes 0x41.

Bridge A1 instead and it's now an address of 0x42. Bridge both A0 and A1 and the address will be 0x43.

Multiple Servos – Controlling the MeArm

In order to demonstrate the use of the PCA9685 PWM module to control multiple servo motors I decided to bring out the MeArm which I built earlier. It has four servo motors.

I connected everything up as follows:



You'll notice that I also added four potentiometers, as before these can be any value of 10k or above and will be used to regulate the operation of each of the four servo motors.

The PCA9685 module hooks up to the SCL and SDA connections on the Arduino. If your Arduino does not have pins for these I2C connections then use analog pin A4 for SDA and pin A5 for SCL.

Note that even if you do have separate SCL and SDA pins you won't be able to use A4 and A5 as analog inputs when using I2C.

The four potentiometers connect to ground on one side and 5 volts on the other. Their wipers connect to analog inputs A0 through A3.

The Arduino power is also used to power the VCC power on the PCA9685 module. A seperate power supply for the four servos is connected to the screw connector on the module.

I connected my servo motors to outputs 0, 4, 8 and 12. You can actually use any four connections, just note them so you can modify the code to match your selection.

As this is the only PCA9685 module I've connected to the Arduino I didn't short out any of the address solder pads.

Now let's look at the sketch I'm using to make this all work:

Arduino PCA9685 4 Motor Controller

```
/*
```

```
  PCA9685 PWM Servo Driver Example
  pca9685-servomotor-demo.ino
  Demonstrates use of 16 channel I2C PWM driver board with 4 servo motors
  Uses Adafruit PWM library
  Uses 4 potentiometers for input

  DroneBot Workshop 2018
  https://dronebotworkshop.com
*/

// Include Wire Library for I2C Communications
#include <Wire.h>

// Include Adafruit PWM Library
#include <Adafruit_PWMServoDriver.h>

#define MIN_PULSE_WIDTH       650
#define MAX_PULSE_WIDTH       2350
#define FREQUENCY             50

Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();

// Define Potentiometer Inputs

int controlA = A0;
int controlB = A1;
int controlC = A2;
int controlD = A3;

// Define Motor Outputs on PCA9685 board

int motorA = 0;
int motorB = 4;
int motorC = 8;
int motorD = 12;

void setup()
{
  pwm.begin();
  pwm.setPWMFreq(FREQUENCY);
}


void moveMotor(int controlIn, int motorOut)
{
  int pulse_wide, pulse_width, potVal;

  // Read values from potentiometer
  potVal = analogRead(controlIn);

  // Convert to pulse width
  pulse_wide = map(potVal, 0, 1023, MIN_PULSE_WIDTH, MAX_PULSE_WIDTH);
  pulse_width = int(float(pulse_wide) / 1000000 * FREQUENCY * 4096);

  //Control Motor
  pwm.setPWM(motorOut, 0, pulse_width);

}

void loop() {

  //Control Motor A
  moveMotor(controlA, motorA);

  //Control Motor B
  moveMotor(controlB, motorB);

  //Control Motor C
  moveMotor(controlC, motorC);

  //Control Motor D
  moveMotor(controlD, motorD);


}
```

The sketch makes use of the Adafruit PWM Servo Driver Library which you will need to install to make this work. It can be installed from the Library Manager in your Arduino IDE.

- Open the Arduino IDE.
- Select *Sketch* from the menu at top.
- Select *Include Library*. A sub-menu will appear.
- Select *Manage Libraries…* from the sub-menu.
- The Library Manager will open.
- Search the Library Manager for “Adafruit PWM”
- The *Adafruit PWM Servo Library Driver* should be the first result.
- Click the *More Info* link to reveal an *Install* button. Use this button to install the library into your IDE.
- Close the library manager.
- The library is now installed and can be used in your IDE.

We begin the sketch by including the Wire library. This is built into your Arduino IDE and is used to control I2C communications.

Next we include the Adafruit PWM Servo Library that we just installed.

We will now define a few constants.

The first two constants define the minimum and maximum pulse width for the PWM signal we will be sending to our servos. A you recall this pulse width will determine the position of the servo shaft.

The third constant we define is the PWM frequency, which for analog servo motors is 50 Hz. If you are using digital servo motors you may want to increase this as they can often use frequencies as high as 200 Hz.

Next we create an object called *pwm* using the Adafruit PWM Library. If you used an address other than the default 0x40 you would need to define it here.

Now we define some variables. The first one is the potentiometer input pins, A0 through A3. After that are the motor outputs on the PCA9685 board, I used 0, 4, 8 and 12 when I hooked up my motors. Change these values if you used different connectors for your motors.

Now onto the Setup. We initialize the *pwm* object we created earlier and then set the frequency of the PWM oscillator to the frequency we defined, which in our case is 50 Hz.

Now we create a function that will drive the motors in response to the potentiometer positions. We can then just call this function for each motor.

Our function is called *moveMotor.* It has two inputs, *controlIn* which represents the potentiometer input and *motorOut* which represents the motor connection on the PCA9685.

The function reads the potentiometer value and converts it to a pulse width. This pulse width is then used with the *setPWM* method of the Adafruit PWM Servo Library to send the pulse to the motor specified by the *motorOut* variable.

In the loop we just call the *moveMotor* function four times, once for each potentiometer-servo motor combination.

The result is that the four servo motors in the MeArm will respond to the potentiometers. In the demo I used slide potentiometers which made it a lot easier to precisely position the MeArm (I use the term “precisely” with a lot of poetic license!).

Conclusion

Servo motors are versatile little devices that have a myriad of uses in hobbyist projects and knowing how to control them is an essential Arduino coding and wiring skill.

Hopefully this article and its associated video have helped shed some light on using servo motors, either connected directly to an Arduino or via I2C using a PCA9685 PWM controller.

So grab yourself a bunch of servo motors and start making things move today!

Resources

Code for this article – The sketches used in the article in one easy to use ZIP file.

Adafruit PWM Servo Library – The Adafruit Library for controlling the PCA9685 16-Channel controller board.

Arduino Servo Library – The Servo library included with the Arduino IDE.