Introduction to the Arduino Kit

Dr. Karim Muci-Kuchler

Introduction

- Arduino is an open source microcontroller platform used for sensing both digital and analog input signals and for sending digital and analog output signals to control devices.
- A microcontroller is a computer chip with minimal processing power that is often used for automatically controlling some type of external device.
- Most microcontrollers contain all their memory and input/output (I/O) peripherical ports on the same chip.
- The Arduino platform adds additional electronics that make it easy to interface the microcontroller to a computer and other devices.

Introduction (Cont.)

- Microcontroller Examples:

Introduction (Cont.)

- The Arduino uses the Atmel ATmega AVR family of microcontroller chips.

- The Atmel ATmega AVR microcontroller is an 8-bit computer with the following built in features:
  - ***Flash memory*** for storing program code statements.
  - ***Static random-access memory (SRAM)*** for storing program data.
  - ***Erasable electronic programmable read-only memory (EEPROM)*** for storing long term data.
  - Digital input and output ports.
  - Analog-to-digital converter (ADC) for converting analog input signals to digital format.

- The ATmega family of microcontrollers has ***very limited memory***.

Introduction (Cont.)

- There are two types of signals that carry information: Analog and digital signals

- **Analog Signal**
  - A continuous electrical signal – can take on any value.
  - This would come from, for example, a sensor that measures the temperature in a room.
  - Analog signals vary in time, and the variations follow that of the non-electric signal that they represent.

- **Digital Signal**
  - A discontinuous electrical signal – can take on only certain values.
  - This might come from a push button, that is open or closed.
  - Digital signals have two discrete amplitude levels:

    0 or 1 / False or True / Low or High.

Introduction (Cont.)

- The language used to write programs for the Arduino derives from the C programming language, with some added features that are unique to the Arduino environment.

- The following web site provides a quick reference for the Arduino programming language:

  https://www.arduino.cc/en/Reference/HomePage

- Programs that you create for the Arduino are called ***sketches*** and the folders where you store the sketches are called ***sketchbooks***.

- Why those names? The Arduino was designed out of necessity by a group of students and instructors looking for a solution to animate their art projects.

Introduction (Cont.)

- The Arduino system was designed for simplicity for non-technical people.
- The software for the Arduino was designed with nonprogrammers in mind.
- Students in non-engineering majors have successfully used the Arduino for projects involving building interactive objects.
- You can find a lot of information about Arduino (including many examples) on the web.
- There are several Arduino models. We will be using the Arduino Uno.

Before we proceed …

- Since we will be working with hardware involving electric circuits, it is important that you gain a basic understanding about them and learn how to work with them safely.
- You are encouraged to find and read on your own references that can help you achieve that goal.
- Here we will only present a little bit of information about working with direct current (DC).
- In DC we will always have a positive (+) pole/terminal and a negative (-) pole/terminal.

Before we proceed … (Cont.)

- The voltage applied to a component and the current that flows through it are two important variables that we need to keep in mind.
- Ground is the reference point in an electrical circuit from which voltages are measured. If a circuit is not grounded, current will not flow.
- When we connect a component, it is often important that we take into account the polarity.
- Always check the voltage/current/power requirements of each component.

Before we proceed … (Cont.)

- Failure to take into consideration the voltage/current/power requirements and/or the polarity when we are connecting a component could result in permanent damage to the component.

Important Acronyms

- Here are some commonly used acronyms that you may want to know.
- I2C = Inter-integrated Circuit (I^2C) Protocol
- Each I^2C bus consists of two signals: SCL and SDA.
  SCL is the clock signal, and SDA is the data signal.
- TX → Transmit
- RX → Receive
- UART = Universal Asynchronous Receiver/Transmitter

Important Acronyms (Cont.)

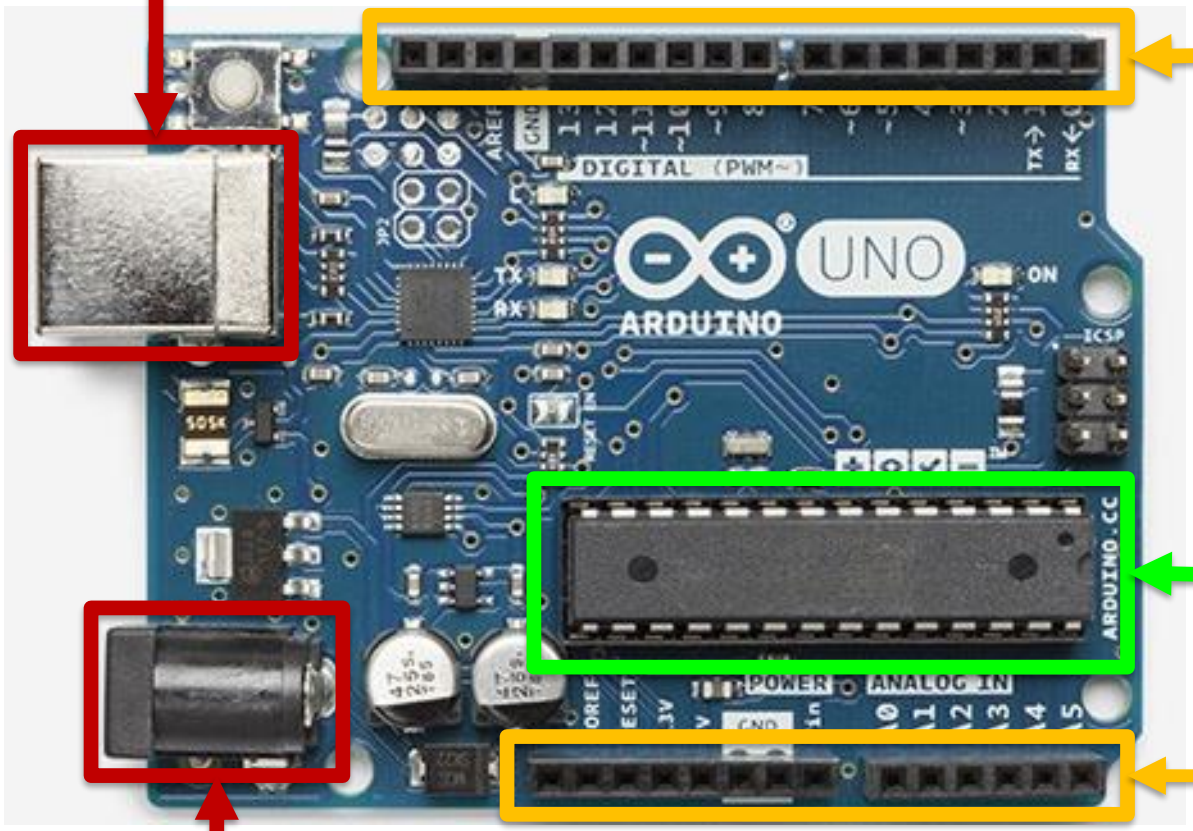
- USB = Universal Serial Bus
- ISP = In-System Programming
- ICSP = In-Circuit Serial Programming
- SRAM = Static Random-Access Memory
- EEPROM = Electrically Erasable Programmable Read-Only Memory
- PWM = Pulse Width Modulation

Important Acronyms (Cont.)

- GND = Ground
- PCB = Printed Circuit Board
- SMD = Surface-Mount Device
- LED = Light-Emitting Diode

Arduino Uno R3

Arduino Uno R3 (Cont.)



Arduino Uno R3 (Cont.)

- Uses an ATmega328 microcontroller.
- ***Header sockets*** make the input and output pins of the microcontroller available to external devices.
- All the microcontroller interfaces can be accessed from the header sockets.
- Header sockets close to the power jack → Analog input interfaces and access to the voltage and ground pins on the microcontroller.
- Header sockets close to the USB interface → Digital I/O interfaces.
- The standard layout of the header sockets allows developers to build plug-in devices called ***shields*** that interface with the Arduino.

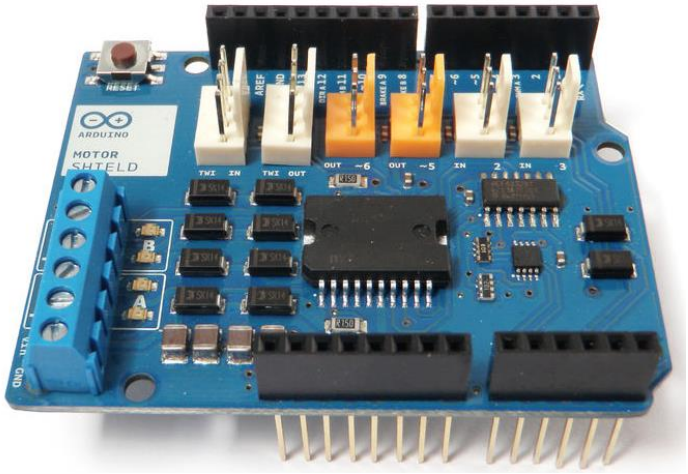
Arduino Uno R3 (Cont.)

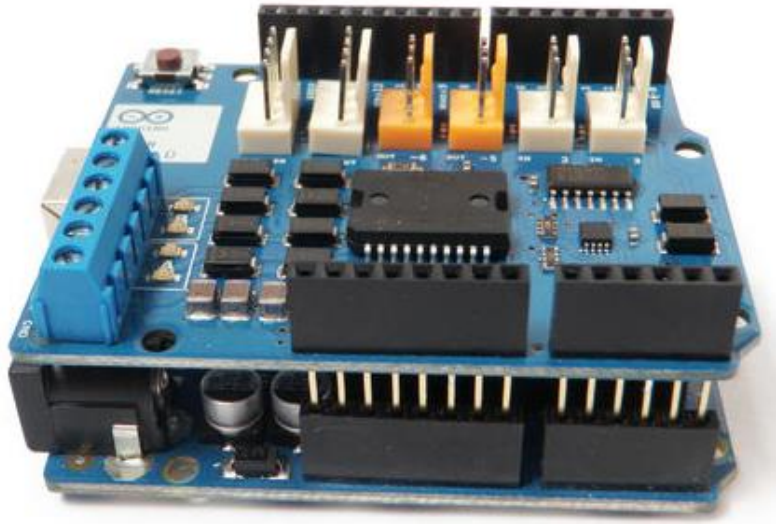
- The board operates at 5V, and has an on-board 3.3V regulator. It can be powered either through USB or with a 7-15V barrel jack power supply.

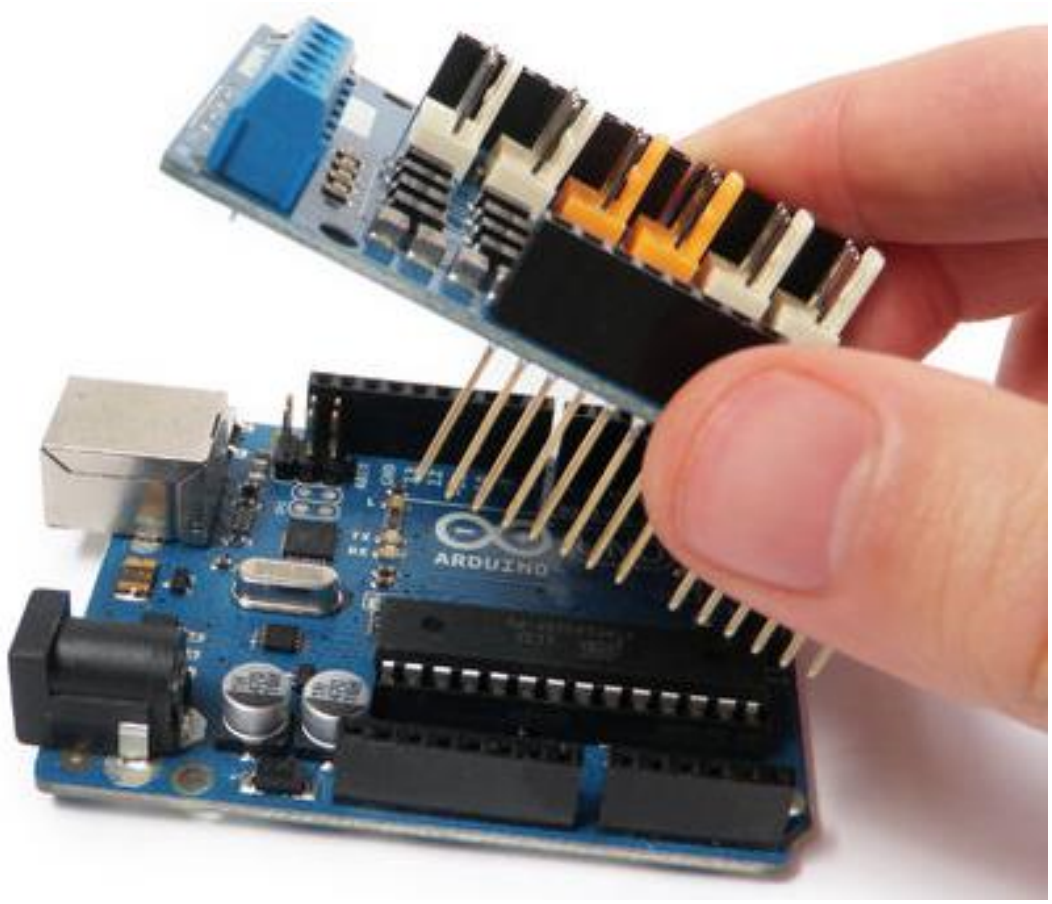
- The Arduino Uno uses a USB type B connector.

- The microcontroller in the Arduino Uno R3 has the following amounts of memory:
  - Flash Memory: 32k bytes
  - SRAM: 2k bytes
  - EEPROM: 1k byte

Arduino Uno R3 – Shield Example



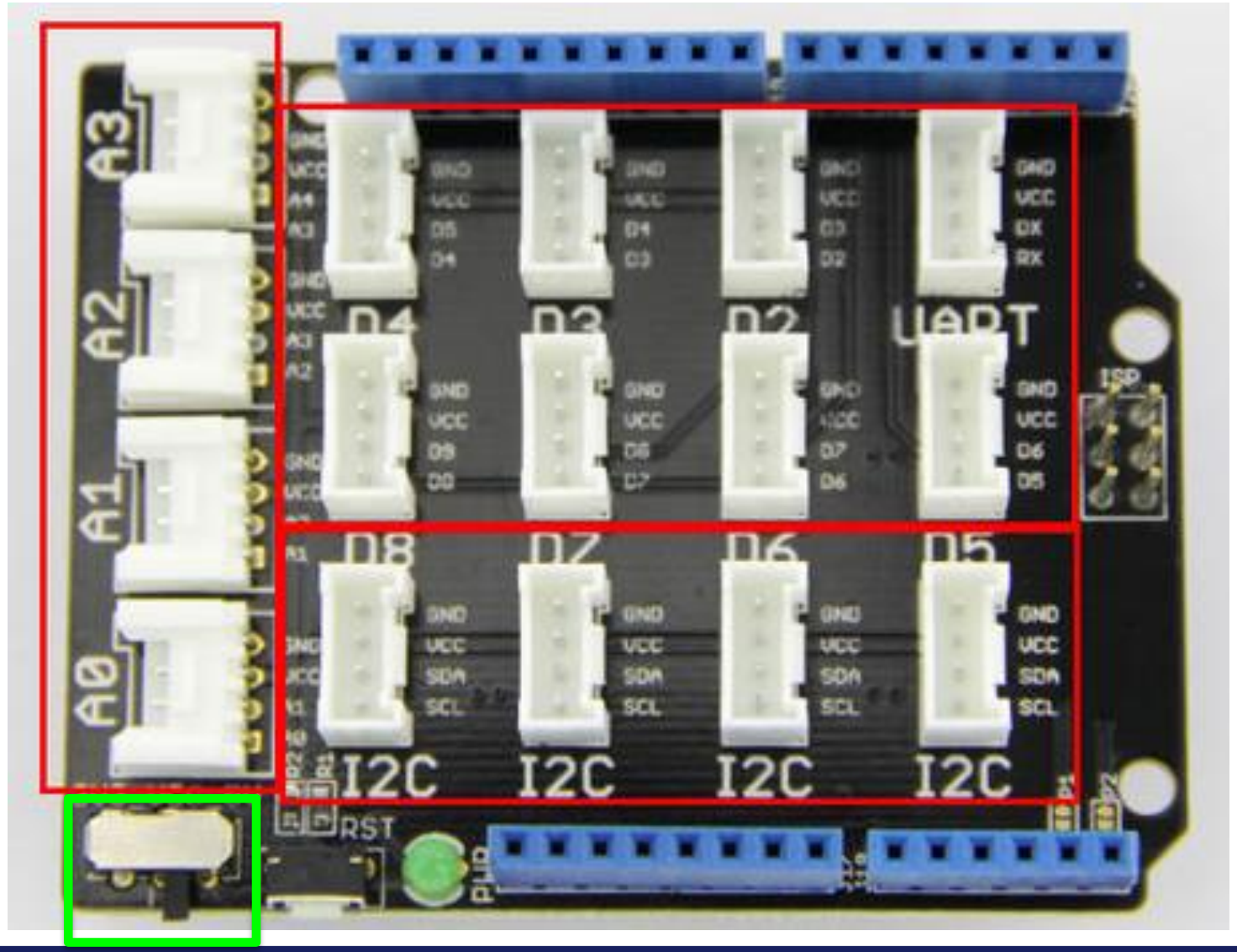




Pictures from: http://www.instructables.com/id/Arduino-Motor-Shield-Tutorial /

Grove Starter Kit for Arduino

Grove Base Shield



Other Shields in the Kit

- In what follows. we will learn about some of the Arduino shields that are part of the kit:
  - **The motor shield**
    Adafruit Motor/Stepper/Servo Shield for Arduino v2
  - **The servo shield**
    Adafruit 16-Channel PWM/Servo Shield
  - **The USB host shield**
    SparkFun USB Host Shield
- Those shields use the I2C communication protocol.

General Information About I2C

- I2C is popular because it uses only 2 wires (SCL and SDA) and multiple devices can share those wires, making it a great way to connect tons of sensors, drivers, expanders, without using all the microcontroller pins.

- When using I2C each device must have a unique address.

- Arduino uses 7-bit notation for addresses. Consequently, the addresses only range from 0 to 127 → from **0x**00 to **0x**7F in hexadecimal (base 16).

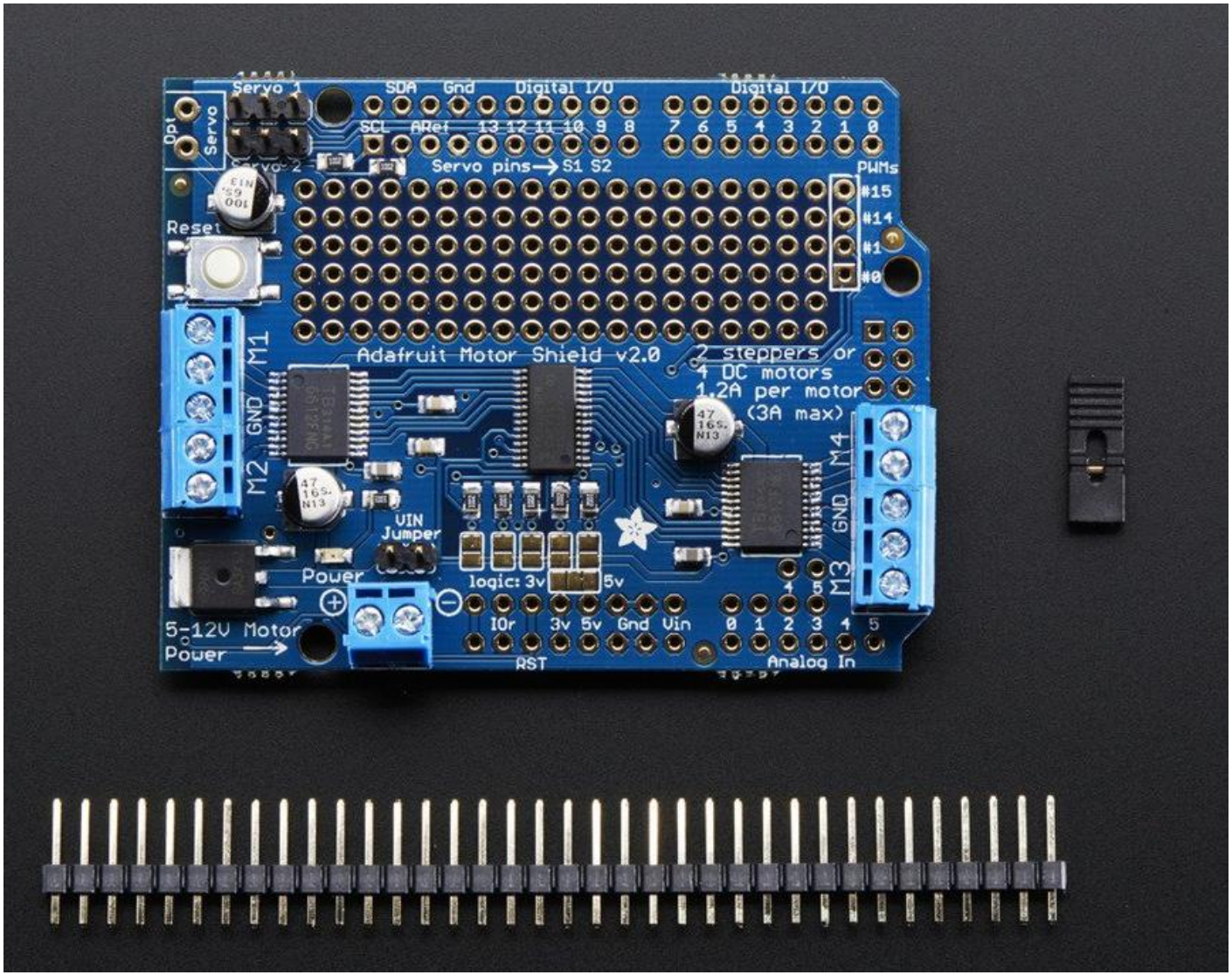
  Note: **0x** is the prefix for hexadecimal numeric constants in computing.

General Information About I2C (Cont.)

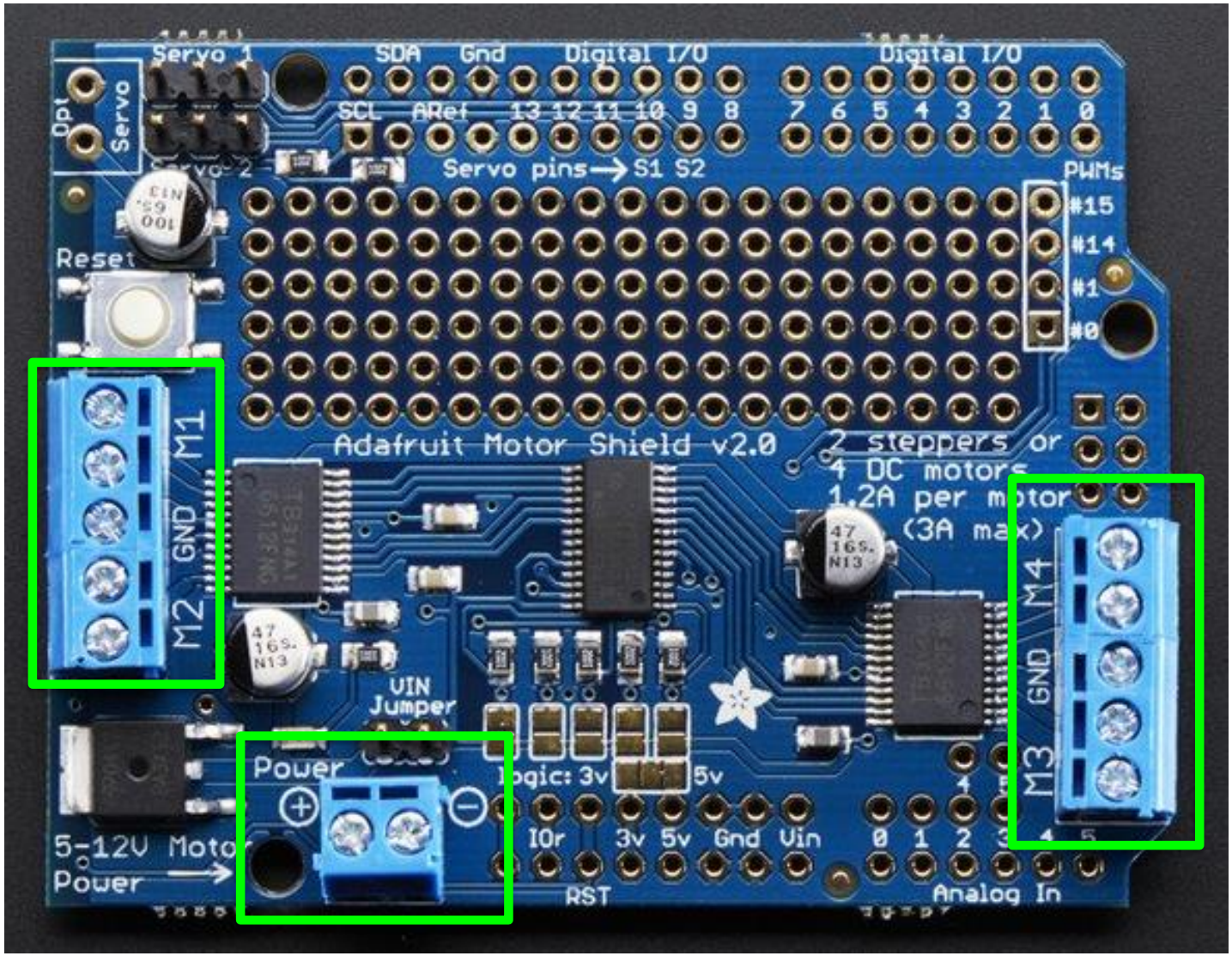
- If you have two devices of the same type and they both have the same address you cannot have both of them on the same I2C lines.

- Some shields/boards have an address-select line or jumper or other configuration. If you can set one to a different address you are good to go.

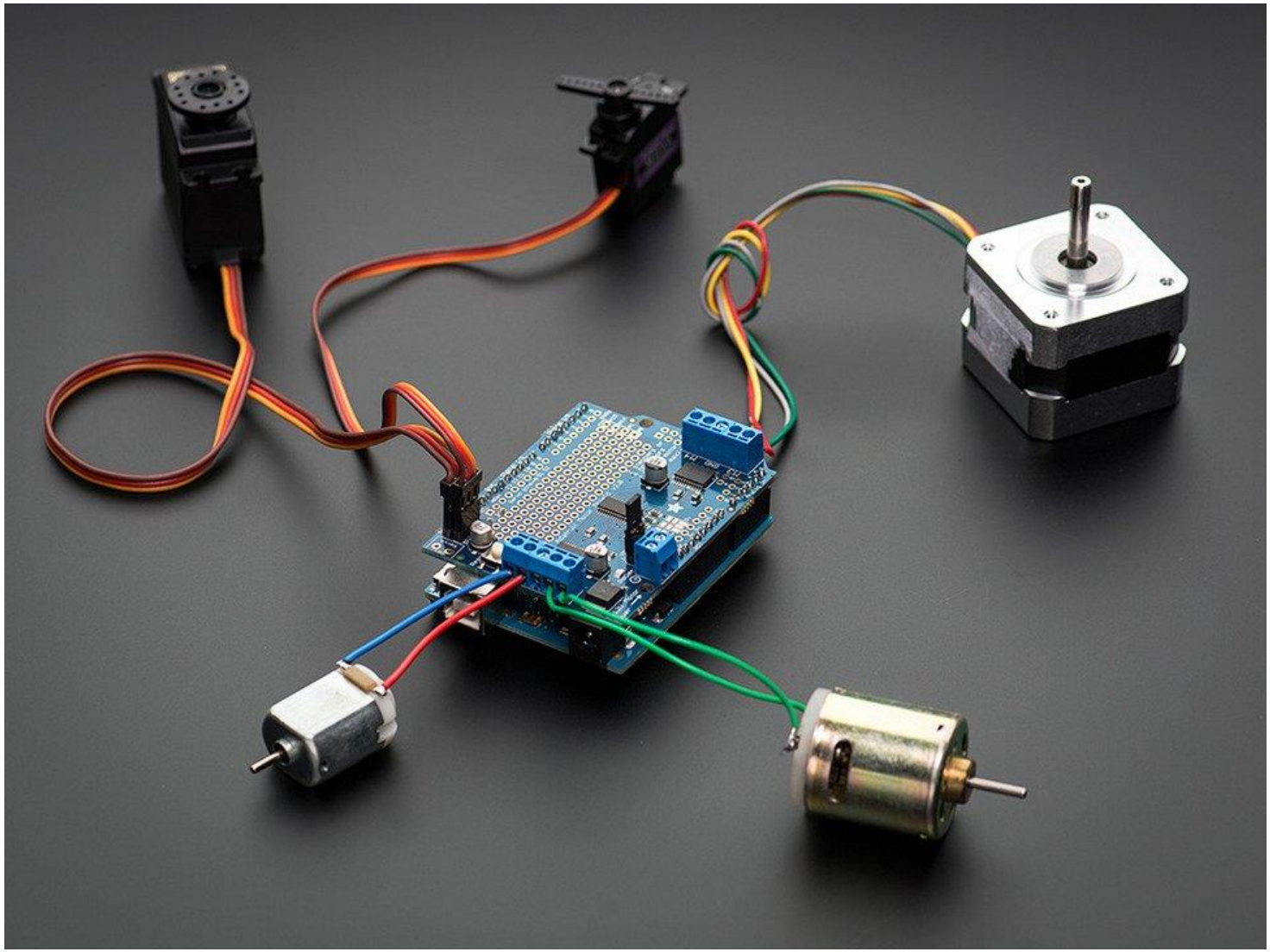
Adafruit Motor/Stepper/Servo Shield for Arduino v2



Adafruit Motor/Stepper/Servo Shield for Arduino v2 (Cont.)



Adafruit Motor/Stepper/Servo Shield for Arduino v2 (Cont.)



Adafruit Motor/Stepper/Servo Shield for Arduino v2 (Cont.)

- 2 connections for 5V “hobby” servos connected to the Arduino's high-resolution dedicated timer.
- 4 H-Bridges: TB6612 chipset provides **1.2A per bridge (3A for brief 20ms peaks) with thermal shutdown protection**, internal kickback protection diodes. Can run motors on 4.5VDC to 13.5VDC.
- **Up to 4 bi-directional DC motors** with individual 8-bit speed selection (so, about 0.5% resolution).
- **Up to 2 stepper motors** (unipolar or bipolar) with single coil, double coil, interleaved or micro-stepping.

Adafruit Motor/Stepper/Servo Shield for Arduino v2 (Cont.)

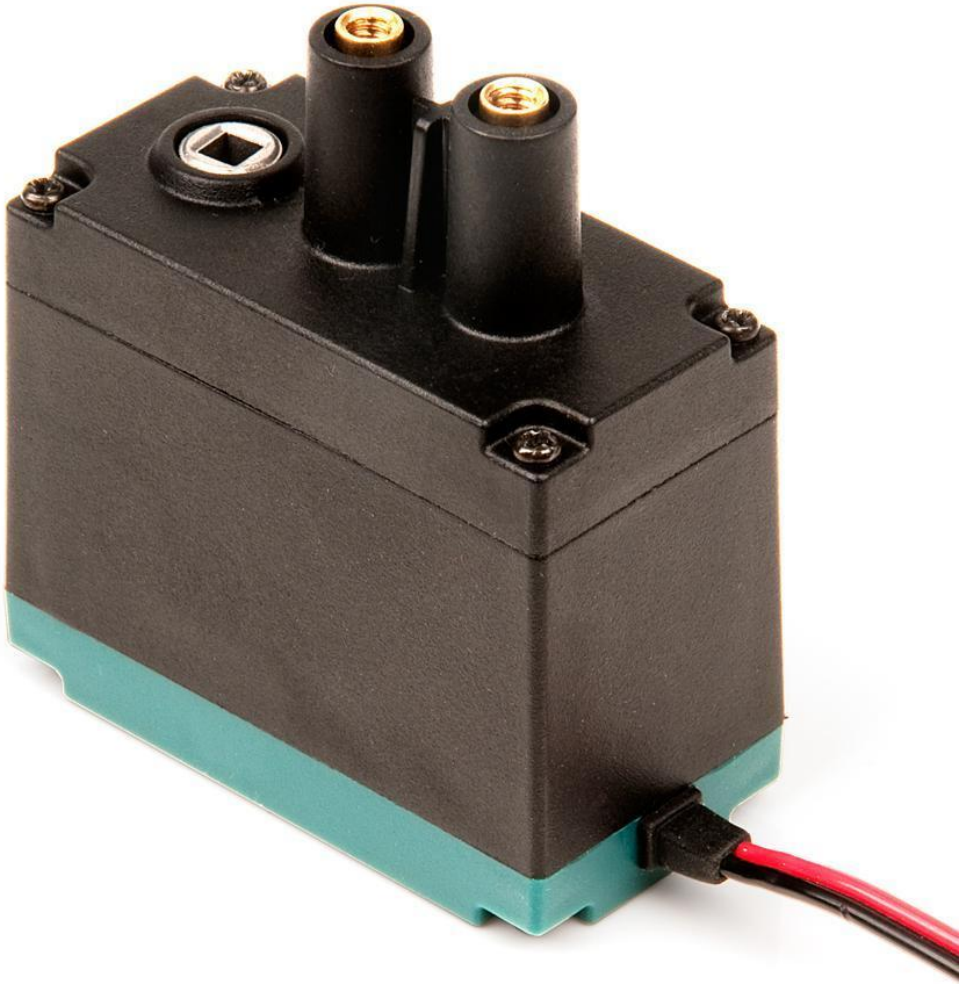
- Motors automatically disabled on power-up.
- Big terminal block connectors to easily hook up wires (18-26 AWG) and power.
- Arduino reset button brought up top.
- Polarity protected 2-pin terminal block and jumper to connect external power, for separate logic/motor supplies.
- You can have up to 32 stackable shields → up to 128 DC motors or 64 steppers.

Adafruit Motor/Stepper/Servo Shield for Arduino v2 (Cont.)

- Stacking motor shields → Use I2C 7-bit addresses between 0x60-0x80, selectable with jumpers.
- Since I2C is a “shared bus” you can still connect other I2C devices to the SDA/SCL pins as long as they do not have a conflicting address.
- Easy-to-use Arduino software library with examples.

DC Motor Example: VEX Motor 393

Part Number: 276-2177



DC Motor Example: VEX Motor 393 (Cont.)

- The 2-Wire Motor 393 is the primary actuator used in the VEX EDR.
- Has steel internal gears.
- Can be configured into a "high speed" version.
- Has an internal PTC that will cut the current being delivered to the motor if they get too hot.
- PTC → Positive temperature coefficient device → A passive electronic component used to protect against overcurrent faults in electronic circuits.

DC Motor Example: VEX Motor 393 (Cont.)

- According to the specifications of the PTC, the device is guaranteed to trip within 7.1 seconds of continuous exposure to 1.8A, and will trip more quickly when exposed to higher currents or is operated above “room temperature”.

DC Motor Example: VEX Motor 393 (Cont.)

- The following specifications are at 7.2V:
  - Free Speed – the fastest possible rotation speed

    As shipped: 100 rpm

    High speed option: 160 rpm
  - Stall Torque – the maximum torque the device can hold

    As shipped: 1.67 N-m (14.76 in-lbs)

    High speed option: 1.04 N-m (9.2 in-lbs)
  - Stall Current: 4.8A – this is high! Don't stall.
  - Free Current: 0.37A

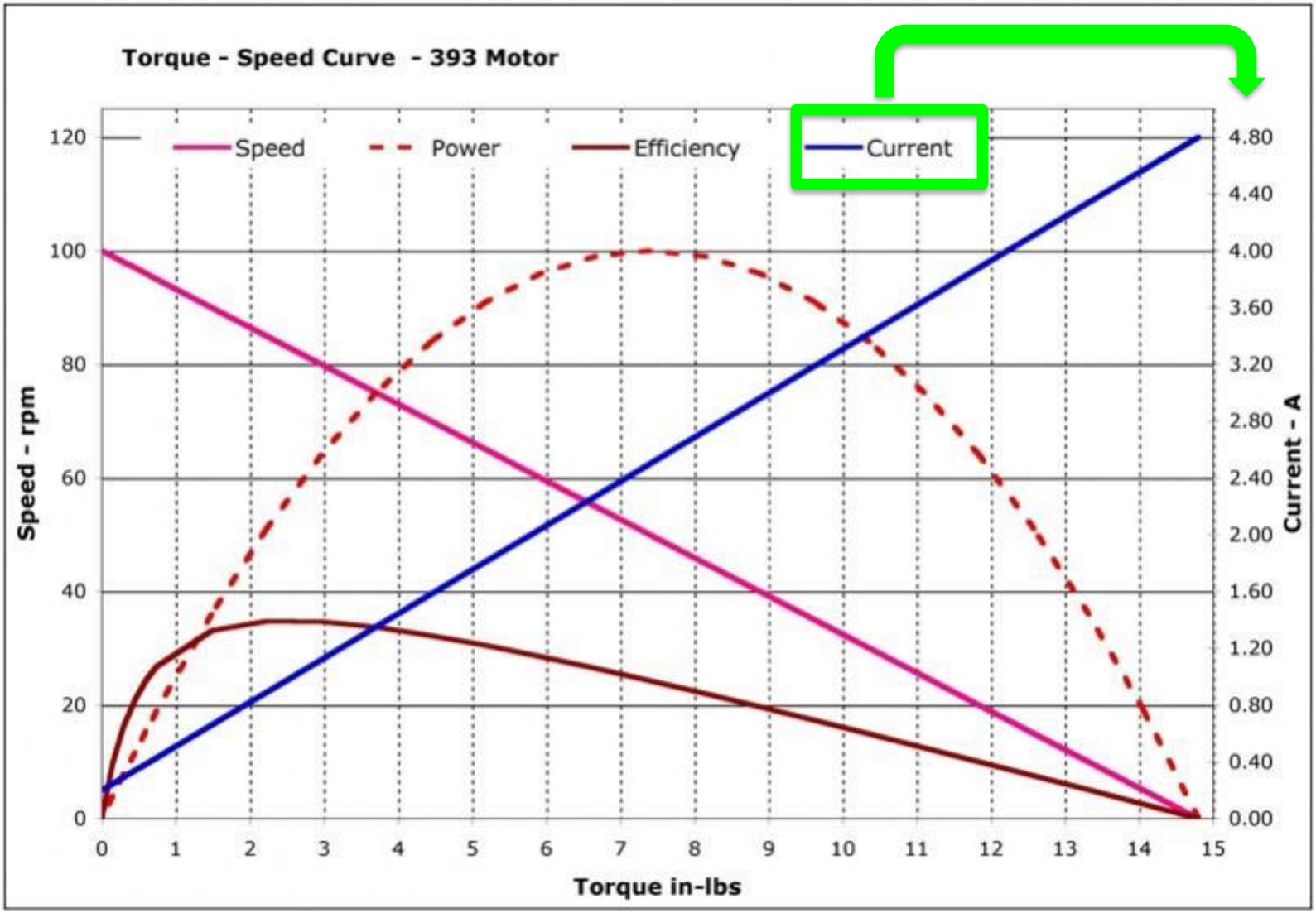
- To learn about DC motors see:

  http://curriculum.vexrobotics.co.uk/curriculum/speed-power-torque-and-dc-motors/dc-motors

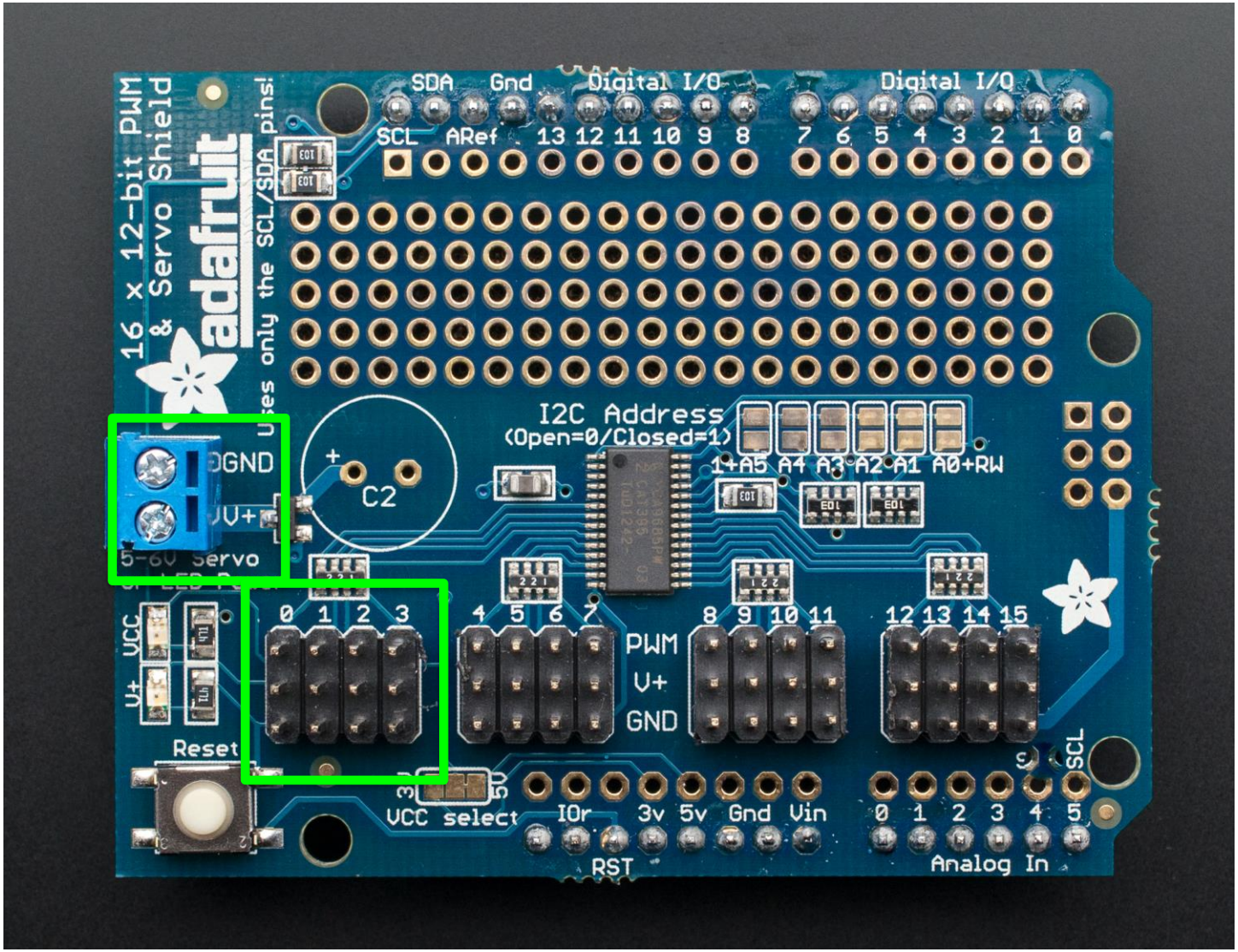
DC Motor Example: VEX Motor 393 (Cont.)

- When you use a DC motor it is a good idea to find information about its performance characteristics.
- One aspect of particular interest is the amount of current that the motor is going to draw at different load/torque levels.
- Keep in mind that there are limitations in the amount of current that the motor shield can provide to each motor.
- A Torque-Speed Curve for the VEX Motor 393 is presented in the next slide. Look at the current that the motor needs at different torque levels and compare with the amount of current that the motor shield can provide and with the specifications of the PTC that the motor has. Can you identify any potential problems?

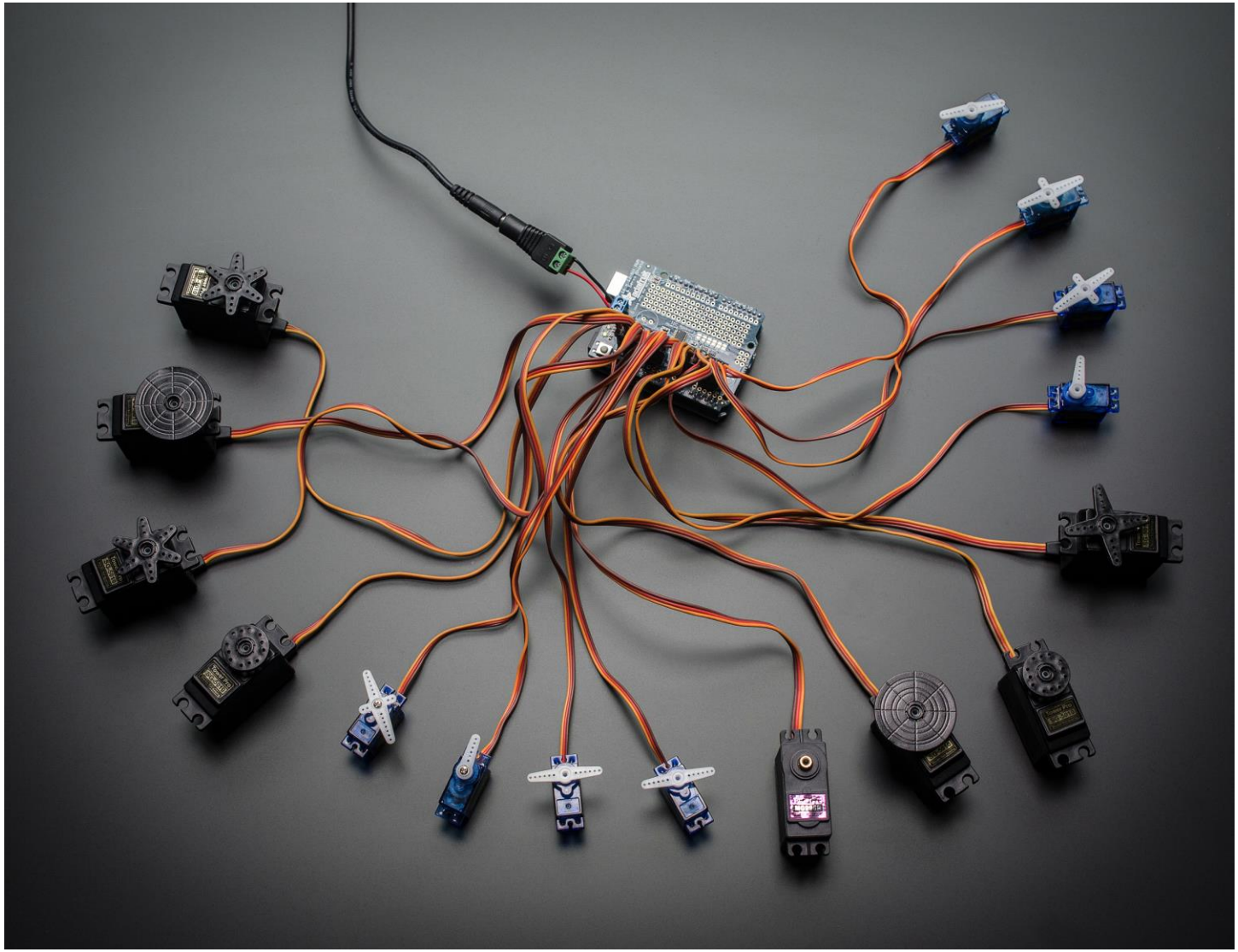
VEX Motor 393 Torque-Speed Curve



Adafruit 16-Channel PWM/Servo Shield



Adafruit 16-Channel PWM/Servo Shield (Cont.)

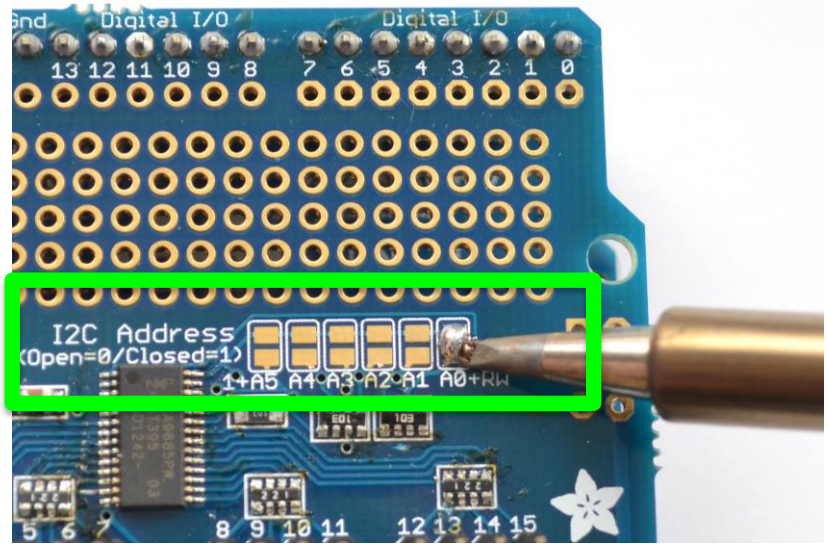


Adafruit 16-Channel PWM/Servo Shield (Cont.)

- The shield plugs in directly into any shield-compatible Arduino.
- The shield can drive up to 16 servos over I2C with only 2 pins.
- The on-board PWM controller will drive all 16 channels simultaneously with no additional Arduino processing overhead.
- You can stack up to 62 of these shields to control up to 992 servos - all with the same 2 pins.
- Each board in the chain must be assigned a unique address. This is done with the address jumpers located on the board.

Adafruit 16-Channel PWM/Servo Shield (Cont.)

- The I2C base address for each board is 0x40.
- The binary address that you program with the address jumpers is added to the base I2C address.
- To program the address offset, use a drop of solder to bridge the corresponding address jumper for each binary “1” in the address.



- **Board 0:**
  Address = 0x40 Offset = binary 00000 (no jumpers required)
- **Board 1:**
  Address = 0x41 Offset = binary 00001 (bridge A0)
- **Board 2:**
  Address = 0x42 Offset = binary 00010 (bridge A1)
- **Board 3:**
  Address = 0x43 Offset = binary 00011 (bridge A0 & A1)
- **Board 4:**
  Address = 0x44 Offset = binary 00100 (bridge A2)
- Etc.

Adafruit 16-Channel PWM/Servo Shield (Cont.)

- The shield has two power supplies.

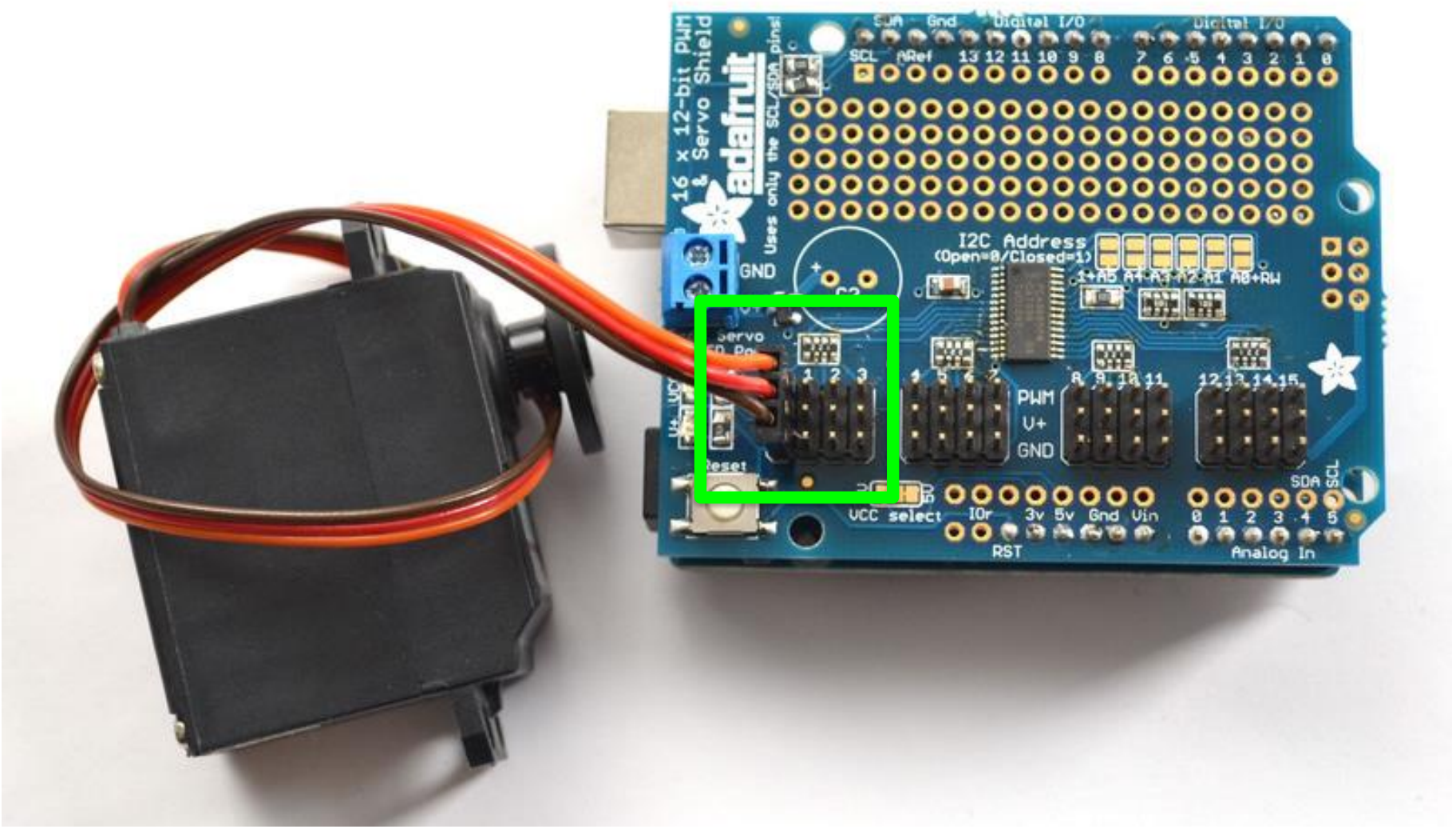
- One power supply is VCC - that is the 5V power from the Arduino. It is used to power the PWM. When this power supply is working you will see a red LED. The red LED must be lit for the Arduino & shield to work!

- To power servos you will need to also connect the V+ power supply - this is the power supply for the servos. **This power supply should be 5 or 6VDC**. You can connect this power through the blue terminal block. There is reverse-polarity protection in case you hook up power backwards. Check the green LED is lit!

Adafruit 16-Channel PWM/Servo Shield (Cont.)

- Nearly all servos are designed to run on about 5 or 6V.
- A lot of servos moving at the same time will need a lot of current.
- Even micro servos will draw several hundred mA when moving.
- Some high-torque servos will draw more than 1A each under load.
- Most servos come with a standard 3-pin female connector that will plug directly into the headers on the Servo Driver.
- Be sure to align the plug with the ground wire (usually black or brown) with the bottom row and the signal wire (usually yellow or white) on the top.

Adafruit 16-Channel PWM/Servo Shield (Cont.)



Adafruit 16-Channel PWM/Servo Shield (Cont.)

- Servo pulse timing varies between different brands and models.
- Since it is an analog control circuit, there is often some variation between samples of the same brand and model.
- For precise position control, you want to calibrate the minimum and maximum pulse-widths in your code to match known positions of the servo.
- Sample code: IDE → File → Examples → Adafruit PWM Servo Driver Library → Servo

Adafruit 16-Channel PWM/Servo Shield (Cont.)

- **pwm.setPWMFreq(*freq*)**
  - ***freq*** → A number representing the frequency in Hz, between 40 and 1000.
  - This function can be used to adjust the PWM frequency, which determines how many full "pulses" per second are generated.
  - This is determined by the manufacturer of the servo you are using and generally should not be changed.

- **pwm.setPWM(*channel, on, off*)**
  - ***Channel*** → The channel for the servo you're moving (0 to15)
  - ***on*** → The tick (between 0 and 4095) when the signal should transition from low to high. Typically this is set to 0.
  - ***off*** → the tick (between 0 and 4095) when the signal should transition from high to low. We'll convert this to desired angle shortly…
  - This function sets the start (on) and end (off) of the high segment of the PWM pulse on a specific channel.

Adafruit 16-Channel PWM/Servo Shield (Cont.)

- **Finding the Minimum:**

  Using the example code, edit SERVOMIN until the low-point of the sweep reaches the minimum range of travel. It is best to approach this gradually and stop before the physical limit of travel is reached.

- **Finding the Maximum:**

  Using the example code, edit SERVOMAX until the high-point of the sweep reaches the maximum range of travel. Again, is best to approach this gradually and stop before the physical limit of travel is reached.

- Use caution when adjusting SERVOMIN and SERVOMAX. Hitting the physical limits of travel can strip the gears and permanently damage your servo.

Adafruit 16-Channel PWM/Servo Shield (Cont.)

- **The Arduino "map()" function** is an easy way to convert between degrees of rotation and your calibrated SERVOMIN and SERVOMAX pulse lengths.

- Assuming a typical servo with 180 degrees of rotation; once you have calibrated SERVOMIN to the 0-degree position and SERVOMAX to the 180 degree position, you can convert any angle between 0 and 180 degrees to the corresponding pulse length with the following line of code:

```
pulselength = map(degrees, 0, 180, SERVOMIN, SERVOMAX);
```

This is 100 for VEX servos

Set this as "off" in pwm.setPWM!

3-Wire Servo Example

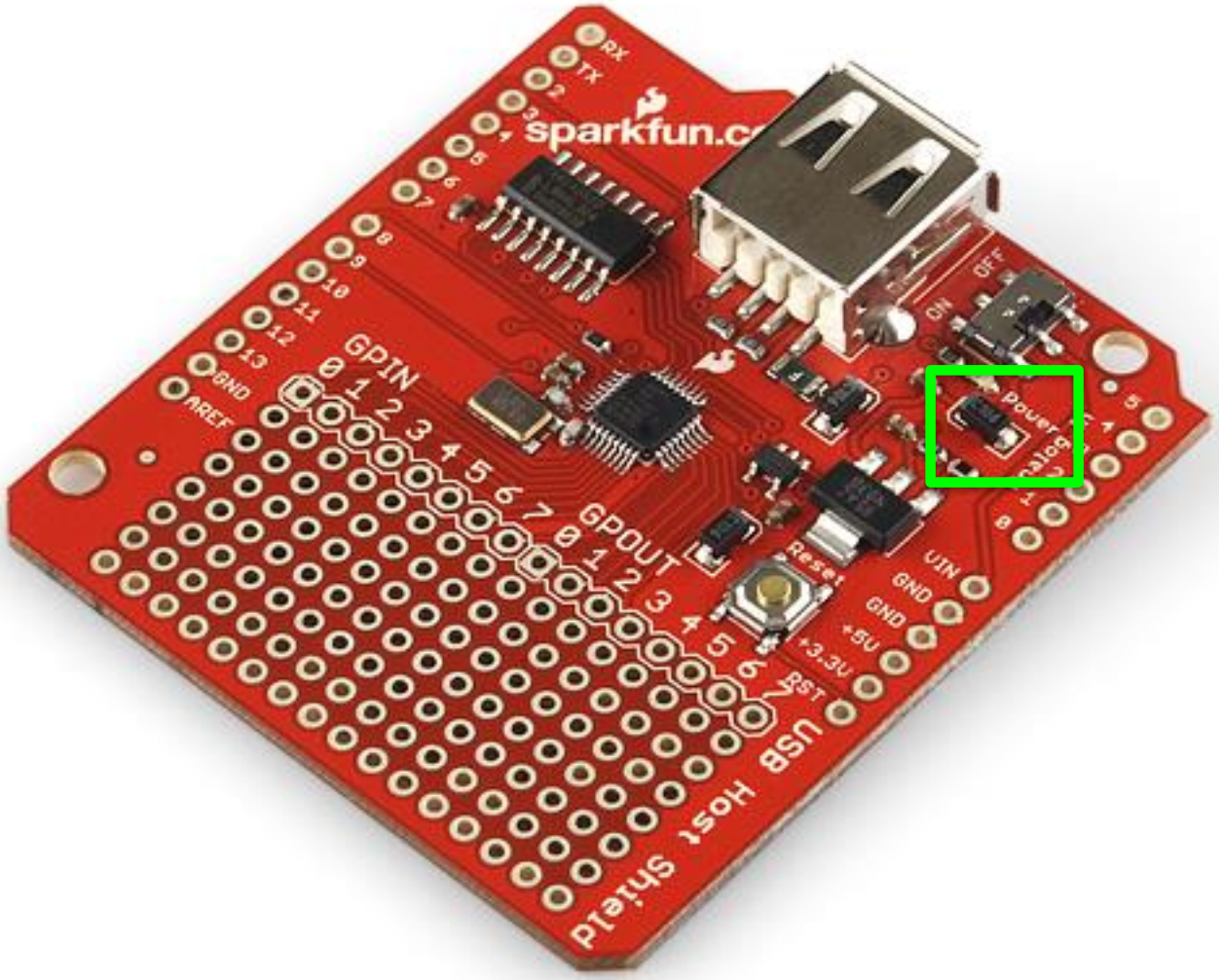
Part Number 276-2162

Servo motors are a type of motor that can be directed to turn to face a specific direction, rather than just spin forward or backward.

3-Wire Servo Example (Cont.)

- **Rotation: 100 degrees**
- Stall Torque: 6.5 in-lbs
- Voltage: 4.4 - 9.1 V (Motor life will be reduced operating outside this range)
- PWM Input: 1ms - 2ms will give full reverse to full forward, 1.5ms is neutral.
- Black Wire: Ground
- Orange Wire: Power
- White Wire: PWM signal
- **Current Draw: 20mA to 1.5 A per Servo**
- Max Power: 4.9 W when rated @ 6V
- Note: Performance varies slightly due to variations in manufacturing.

SparkFun USB Host Shield



SparkFun USB Host Shield (Cont.)

- The SparkFun USB Host Shield contains all of the digital logic and analog circuitry necessary to implement a full-speed USB peripheral/host controller with your Arduino.
- This means you could use your Arduino to interface with and control any USB slave device like a Bluetooth dongle.
- The shield connects the Arduino's hardware SPI pins (D10-13) to the MAX3421E. A USB type A female connector is wired up to the IC, and it also supplies 5V as any normal USB port would.
- The Host Shield takes its power from the "Vin" pin on the Arduino. Power from that pin is regulated to both 5V and 3.3V on the shield.

Bluetooth 4.0 USB Module (v2.1 Back-Compatible)

Bluetooth 4.0 USB Module (Cont.) (v2.1 Back-Compatible)

- CSR8510 A10 Bluetooth USB host.
- This adapter is backwards compatible with v2.1 and earlier, but also supports the latest v4.0/Bluetooth Low Energy. Inside lies a CSR8510 Bluetooth USB host
- Works on a Windows 7 computer installing as a “generic” Bluetooth device.
- Has a unique MAC (Media Access Control) address.
- Drivers for CSR Bluetooth modules are built into Windows XP/7/8/10 so you do not need to install any software or download anything.

SONY PS3 DualShock 3 Wireless Controller

SONY PS3 DualShock 3 Wireless Controller (Cont.)

- The current setup may not work with wireless PS3 controllers not made by SONY.

- You need to find the MAC address of the Bluetooth dongle and then use the program **SixaxisPairTool** that can be found at http://dancingpixelstudios.com/sixaxis-controller/sixaxispairtool/ to load that MAC address to the PS3 controller.

- We have noticed that the PS3 controller will not charge when connected to some USB ports. Check as soon as possible to see if you can charge it with your laptop or with other computer that you may have access to.

References

- Blum, Richard. Sams Teach Yourself Arduino Programming in 24 Hours. Pearson Education, Inc. 2015.
- https://en.wikipedia.org/wiki/Arduino
- https://www.arduino.cc/
- http://www.seeedstudio.com/depot/Grove-Starter-Kit-for-Arduino-p-1855.html
- https://www.adafruit.com/product/1438
- https://learn.adafruit.com/adafruit-16-channel-pwm-slash-servo-shield/overview?view=all
- http://www.vexrobotics.com/motors.html
- https://www.vexrobotics.com/wiki/2-Wire_Motor_393
- http://curriculum.vexrobotics.co.uk/curriculum/speed-power-torque-and-dc-motors/dc-motors

References (Cont.)

- https://www.vexrobotics.com/wiki/3-Wire_Motor
- https://www.sparkfun.com/products/9947
- https://www.adafruit.com/products/1327