Advanced Fundamentals

DC Motors with EZB Microcontroller

All EZB's are microcontrollers, this includes EZ-Robot EZ-B v4 and Arduino. Microcontrollers are computer chips or integrated circuits. Those integrated circuits do not have the "power" to provide current to things like electric motors and relays. That is why Transistors and H-Bridges were invented. A Transistor can provide a large amount of current which enables things like DC Motors to be powered.

In the case of connecting a microcontroller to a DC motor, a device called an H-BRIDGE (sometimes called Motor Controller) is necessary.

An H-Bridge is an electronic circuit that enables a voltage to be applied to a motor in either direction. It's triggered by TTL signals from a microcontroller, such as any EZ-B. These circuits are often used in robotics and other applications to allow DC motors to run forward and reverse.

You can purchase H-Bridges of various types that each provides a different amount of current. The current is how many AMPS a DC motor requires.

2 Channel 2.5 amp HBridge from EZ-Robot
A Dual H-Bridge (such as the EZ-Robot 2.5 Amp H-Bridge) controls 2 motors and can be connected to an EZ-B by 6 digital signal wires. 2 wires control each motor, and 2 wires are for PWM. PWM signals control the speed of each motor.

- The most common H-Bridge for an average-sized 2.5 amp DC motor is available here: 2.5 Amp Motor Controller - Products - EZ-Robot (ez-robot.com)
- Here is the ARC robot skill that explains a connection: Dual H-bridge W/PWM - Movement Panels - Skill Store - Products - Synthiam

Sabertooth HBridge Motor Controllers
Sabertooth HBridges come in a variety of types for amperage for different applications. The great advantage to a Sabertooth motor driver is they require fewer wires than a general H-Bridge, such as the EZ-Robot one.
- They are available on their store here: Dimension Engineering - R/C, Power Electronics, Sensors
- The ARC robot skill for the Sabertooth is here: Sabertooth Movement Panel - Movement Panels - Skill Store - Products - Synthiam

Use DC Motor As Servo

A servo is a type of device used in the world of electronics and robotics for precise control of motion. It consists of a motor (typically a DC or AC motor), a control circuit, and a feedback system, usually a potentiometer. The key feature of a servo is its ability to control its output shaft's position accurately. This is achieved by sending a specific signal to the servo, which moves the shaft to a corresponding angle. Servos are commonly used in applications like radio-controlled models, robotics, and precision mechanical systems because they provide controlled movement and positioning.

While this tutorial demonstrates one solution, you may also consider other options such as

using products specifically designed for this. Such as the Dimension Engineering Sabertooth.

Why Would You Need a DC Motor as a Servo?

Using a DC motor as a servo hinges on harnessing its power and controllability. DC motors are appreciated for their simple design, high efficiency, robust performance, and ability to produce high torque at low speeds. However, unlike standard servos, they lack precise control of position. By integrating a DC motor with a control system similar to a servo, which includes feedback mechanisms and an H-bridge for direction control, it becomes possible to use the "infinitely powerful" nature of the DC motor with the precision and accuracy of a servo.

This combination is particularly beneficial in applications requiring high power and precise control. For instance, in industrial robotics or heavy-duty mechanical systems, where the torque and speed of a DC motor are necessary, but so is the precise positioning capability of a servo, this hybrid approach offers the best of both worlds. It allows for creating a powerful and precise system, expanding the possibilities in engineering and robotics.

Objective

To transform a standard DC motor into a precision-controlled device akin to a servo. This is achieved by repurposing the PCB from an existing servo and integrating it with an H-Bridge, which controls the motor's direction and speed.

Key Points and Connection Details

- The servo PCB is powered by +3V or +5V and is aligned with the logic input levels of the H-Bridge.
- Components (PCB and potentiometer) are salvaged from an existing servo.
- The DC motor receives its power independently, allowing it to operate at its full capacity within the limits of both the motor and the H-Bridge.
- The motor wires from the servo PCB are connected to the INA and INB inputs of the H-Bridge, allowing the servo PCB to control the H-Bridge and, in turn, the motor directly.

Components Required

- DC Motor
- Repurposed Servo PCB
- H-Bridge Circuit
- Salvaged Potentiometer
- Separate Power Supplies (Servo PCB can be powered by +3 or +5 from the microcontroller, HBridge motor can be powered directly from the battery)
- Connecting Wires and Accessories

Process Overview

1. **Extracting Servo Components:** Dismantle an existing servo to obtain the PCB and potentiometer.
2. **H-Bridge Setup:** Connect the DC motor to the H-Bridge.
3. **Servo PCB to H-Bridge Connection:**

Identify the motor output wires on the servo PCB (typically two wires used to control the original servo motor).

Connect these wires to the INA and INB inputs of the H-Bridge. This setup allows the servo PCB to send directional control signals (forward or reverse) to the H-Bridge, which controls the DC motor.

4. **Integrating the Potentiometer:** Attach the potentiometer to the DC motor for position feedback.
5. **Powering the System:** Use a +3V or +5V supply for the Servo PCB and a suitable separate supply for the motor and H-Bridge.

Applications and Advantages

This approach is beneficial for projects needing the power of a DC motor with servo-like precision control. The method is cost-effective, resource-efficient, and allows for high customization. By utilizing separate power sources for the control circuit and the motor, the setup ensures operational safety and maximizes the motor's performance potential.

Troubleshooting

If the motor is spinning in the opposite direction, reverse the INA and INB wires between the Servo PCB and HBridge

Robot with Wheel Steering

How to Control a Robot with Wheel Steering (Car-Like Design)

Robots with wheel steering, similar to a car, are less common in robotics due to limitations in agility. However, it’s entirely possible to implement this design using Synthiam’s ARC software and the **Custom Movement Panel v2**. This tutorial will guide you through the process, highlight key considerations, and explain the limitations of this approach.

Why Caster Wheels Are Preferred

Wheel-steered robots lack the ability to pivot in place, which impacts their agility. This makes them less suitable for navigating tight spaces or complex environments. Instead, designs with caster wheels or pivoting mechanisms allow for smooth directional changes and reduce strain on motors or servos. Nonetheless, if you opt for a car-like wheel steering design, here’s how you can make it work effectively.

What You Need

- A robot chassis with wheel steering.
- Synthiam ARC software.
- Custom Movement Panel v2 (available here).
- Motor controller and servos connected to the robot’s hardware.

Steps to Implement Wheel Steering

1. Set Up the Custom Movement Panel

Add the **Custom Movement Panel v2** to your ARC project and configure the ports for:

- Forward motor control.
- Reverse motor control.
- PWM (speed control).
- Servo (steering).

2. Understand the Movement Manager

In ARC, the **Movement Manager** controls all movement panels. When a robot skill requests to "move forward," it specifies left and right wheel speeds. For wheel steering:

- Forward and reverse motions can include slight turns (differential speeds).
- Steering adjustments are based on these speed differences.

---

3. Write Control Scripts

Forward Script

```
// Set the motor direction to forward
Digital.set(d0, true); // Enable forward direction on motor
Digital.set(d1, false); // Disable reverse direction on motor

// Calculate the speed for the PWM
var topSpeed = Math.max(Movement.getSpeedLeft(), Movement.getSpeedRight());
var speed = Utility.map(topSpeed, 0, 255, 0, 100);
PWM.set(d2, speed);

// Calculate the speed difference
var diffSpeed = Movement.getSpeedLeft() - Movement.getSpeedRight();
var steeringAngle = Utility.map(diffSpeed, -255, 255, 1, 180);
Servo.setPosition(d3, steeringAngle);
```

Reverse Script

```
// Set the motor direction to reverse
Digital.set(d0, false); // Disable forward direction on motor
Digital.set(d1, true); // Enable reverse direction on motor

// Calculate the speed for the PWM
var topSpeed = Math.max(Movement.getSpeedLeft(), Movement.getSpeedRight());
var speed = Utility.map(topSpeed, 0, 255, 0, 100);
PWM.set(d2, speed);

// Calculate the speed difference
var diffSpeed = Movement.getSpeedLeft() - Movement.getSpeedRight();
var steeringAngle = Utility.map(diffSpeed, -255, 255, 1, 180);
Servo.setPosition(d3, steeringAngle);
```

Stop Script

```
// Disable motor movement
Digital.set(d0, false); // Disable forward direction on motor
Digital.set(d1, false); // Disable reverse direction on motor
PWM.set(d2, 0); // Disable PWM
```

---

4. Test and Calibrate

- Assign ports (`d0`, `d1`, `d2`, `d3`) to match your robot's wiring.
- Test each script to ensure proper behavior.
- Adjust servo angle ranges in `Utility.map` based on your servo's specifications.

5. Limitations of Wheel Steering

- **Reduced Maneuverability**: Without a pivoting mechanism, tight turns are challenging.
- **Increased Servo Strain**: Continuous adjustments during steering can wear out servos faster.
- **Navigation Restrictions**: Difficult to handle complex environments or obstacles.

For better performance, consider using caster wheels or designs inspired by robot vacuums.

Additional Notes

- Forward and reverse movements can include slight directional adjustments using speed differentials.
- The **Movement Manager** cannot handle pure left or right turns in this setup.
- Ensure proper power supply to avoid hardware failures.

By following these steps, you can control a robot with wheel steering effectively in ARC. While this design has its drawbacks, it's a functional and interesting way to experiment with robotics.

Finger Mechanisms For Servo Protection

Let's discuss designing a robot hand finger mechanism with mechanical and electronic compliance strategies to protect servos. Robotic hands frequently rely on small hobby servos to articulate individual fingers and perform grip actions. Because many of these servos do not support torque sensing, current monitoring, or stall detection, they can easily become overloaded when gripping objects. This can lead to stripped gears, overheating, premature wear, high power consumption, and unpredictable grip forces.

This article outlines multiple engineering approaches — mechanical and electronic — to protect servos from excessive load. Whether designing a simple 3D-printed gripper or a complex humanoid hand, these solutions help improve reliability, lifespan, and performance when used with Synthiam ARC.

Why Mechanical Compliance Is Critical

In the absence of torque feedback, a typical servo will attempt to reach its commanded position even when the finger cannot move further. When gripping an object, this can cause:

- Excessive torque leading to gear damage
- Stall conditions that overheat motors
- High battery drain
- Reduced servo lifespan over repeated cycles
- Inconsistent grip force

Mechanical compliance — a way for the system to "give" under stress — is one of the most effective methods to prevent damage. There are many ways to implement compliance, depending on design constraints and performance goals.

Mechanical Compliance Solutions

Below are multiple approaches you can integrate into a servo-driven finger design. Many can be mixed or layered for additional protection.

---

1. Magnetic Slip Clutch

This method uses two concentric plates with embedded magnets. Under normal operation, magnetic attraction keeps the plates coupled, transmitting torque. When the load exceeds a threshold, the plates slip, preventing servo overload.

Pros

- Excellent overload protection
- Low mechanical wear
- Smooth and predictable slip behavior

Cons

- Finger position becomes out of sync after slipping
- Requires mechanical stop at "fully open" for auto-realignment
- Larger axial space required

Tip: Add a ControlCommand script to periodically re-align the fingers by commanding the hand to fully open.

---

2. Flat Coil (Disc-Style) Torsion Spring

This compact disc-shaped spring functions like a clock spring. The inner hub connects to the servo; the outer ring connects to the finger. When gripping, the spring absorbs torque by twisting slightly.

Pros

- Compact and thin profile
- Maintains finger alignment (elastic deformation only)
- Smooth, gentle grip pressure

Cons

- Limited torque absorption range
- 3D-printed versions may fatigue over time

Works best in tight spaces such as humanoid hand knuckles or finger bases.

---

3. Helical Beam Coupler

A helical coupler (sometimes called a "beam coupler") consists of a solid cylinder with a laser-cut or CNC-cut spiral pattern. This creates a flexible torsion section that can bend, twist, and absorb shock loads.

Pros

- Can flex in torsion, angular offset, and slight axial compression
- Maintains rotational connection without slipping
- Printable or easily purchased

Cons

- Longer axial length than a disc spring
- Torque capacity depends on material and cut geometry

Ideal for finger joints that require both torsional and slight misalignment compensation.

---

4. Compliant Finger Pads and Gripper Surfaces

One of the simplest strategies is to add compression material where the finger contacts the object. This was used on the EZ-Robot JD grippers using foam inserts.

Materials include:

- High-density EVA foam
- Rubber pads
- Silicone inserts
- TPU-printed flexible pads

Pros

- Easy to implement
- Improves grip stability and traction
- Reduces peak load on servos

Cons

- Limited torque absorption
- Does not protect from full stalls

---

5. Slip Gears / Cam-Over (Detent) Mechanisms

In a cam-over gear, the gear "jumps" teeth or disengages when the torque exceeds a threshold. This protects the servo by mechanically limiting force.

Pros

- Strong overload protection
- Simple mechanical concept

Cons

- Produces audible clicks when triggered

- Finger will lose tracking until reset

This is commonly used in consumer torque-limited screwdrivers and some industrial grippers.

---

6. Tendon-Driven Fingers with Elastic Back-Tension

Instead of directly mounting fingers to servo horns, many hands use tendons (cables) to pull the fingers closed. An elastic or spring-loaded mechanism pulls the fingers open.

Pros

- Cables naturally slip when overloaded
- Servos are isolated from impact loads
- Allows very compact finger joints

Cons

- Cable stretch must be calibrated
- Finger position tracking requires careful tensioning

This mimics biological tendons, making it popular in humanoid robotics.

---

7. Soft Robotics Finger Elements

Soft robotics uses silicone, TPU, or pneumatic structures instead of rigid linkages. These are inherently compliant and protect servos by absorbing forces.

Pros

- Virtually impossible to damage a servo through gripping
- Human-safe and forgiving
- Conforms naturally around objects

Cons

- Lower precision without sensors
- Requires entirely different design philosophy

A hybrid rigid/soft design can combine precision with safety.

Electronic and Software-Based Protection

Mechanical solutions are ideal, but electronics and firmware can also help prevent overload.

---

8. Smart Servos with Torque Sensing

Upgrading to "smart servos" provides built-in torque feedback and overload protection. Many digital servo ecosystems (e.g., Dynamixel, LewanSoul/HiWonder LX series, Robotis, Hitec D-series, etc.) offer:

- Torque limiting
- Temperature monitoring
- Stall detection and shutdown
- Current monitoring
- Closed-loop position control with feedback

Pros

- No need for mechanical clutches
- Precise grip force control
- Highly reliable and long-lasting

Cons

- More expensive
- Sometimes larger than hobby servos

Synthiam ARC supports many smart servo families directly or through robot skill plug-ins.

---

9. External Current Limiting or Servo Drivers

Adding a current-limiting driver board or using custom electronics can prevent servo burnout by reducing voltage or cutting power during overload.

Examples:

- Electronic current-limited servo controllers

- MOSFET-based stall detectors
- Thermal fuses on servo power lines

While not as precise as torque sensors, these methods still protect the servo during severe stalls.

---

10. Software-Based Grip Control in ARC

Even without special hardware, Synthiam ARC can improve servo safety through scripting and timing.

Recommended practices:

- Use timed gripping: close the finger for a fixed duration rather than commanding a full rotation.
- Insert small delays between incremental position commands.
- Use `ServoRelease` after completing grip actions to prevent constant force loading.
- Create safe zones, limiting fingers to positions that do not cause servo stall.

Software mitigation does not replace mechanical compliance, but it further reduces wear.

Comparison Summary

Solution	Alignment Kept?	Protection Level	Complexity	Best Use Case
Magnetic Slip Clutch	Sometimes (requires reset)	High	Medium	High-force robotic hands
Flat Disc Torsion Spring	Yes (elastic only)	Medium	Low	Compact humanoid fingers
Helical Beam Coupler	Yes	Medium	Low/Medium	Fingers needing alignment flexibility
Compliant Pads	Yes	Low	Very Low	Simple grippers
Tendon + Elastic System	Yes	Medium	Medium	Human-like hand articulation
Smart Servos	Yes	Very High	Medium/High	Professional robotic hands

---

Final Recommendations

Every robotic hand benefits from mechanical compliance to protect servo motors. For most 3D-printed hands, we recommend:

- **Flat disc springs** for compact compliance and alignment.
- **Helical couplers** for flexible and forgiving finger movement.
- **Magnetic clutches** for high load applications where slippage is acceptable.
- **Smart servos** if budget allows, providing built-in torque control.
- **Compliant finger pads** as a simple enhancement for grip comfort.

Combining multiple approaches often yields the best results. For example: a tendon-driven finger with a flat disc spring and compliant pads will have excellent durability, accuracy, and protection.

Before publishing STL files or deploying a robotic hand in a real environment, incorporating at least one overload-protection system is strongly recommended.

IO Port Types

Every Synthiam ARC-powered robot relies on ports to sense and interact with the world. Ports are the physical connection points on any Synthiam ARC EZB controller/firmware where hardware devices plug in, such as servos, sensors, LEDs, buttons, cameras, and motor controllers. Without ports, your robot would have no way to move, detect objects, respond to input, or communicate with other electronics.

In simple terms, a **port** is how information and power flow between your robot's brain (the EZ-B) and the real world. When ARC sends a command, it travels through a port to a device. When a sensor detects something, the information is returned to ARC via a port. Understanding ports is one of the most essential fundamentals of robotics, because every action your robot performs depends on them.

Ports are often described using the term **I/O**, which stands for **Input / Output**.
Input means data flowing *into* the EZ-B (such as a sensor reading or button press).
Output means data or power flowing *out of* the EZ-B (such as moving a servo, turning on an LED, or sending data to another device). Some ports can do both input and output, while others are designed for only one direction.

Different devices require different types of signals. Some devices require only an on/off signal, while others require precise voltage measurements or high-speed digital communication. Because of this, the EZ-B provides several **port types**, each optimized for a specific task. ARC makes working with these ports easy by abstracting the low-level electronics, letting you focus on your robot's behavior rather than the raw wiring details.

The sections below explain each port type in detail: what they are used for, how they behave electrically, and the types of devices commonly connected to them. If you are new to robotics, read this carefully—understanding ports will make everything else in ARC much easier to learn.

Digital

Digital ports work with signals that are either **True (On)** or **False (Off)**. There is no in-between value. A True value indicates a voltage above approximately 1V, while a False value indicates a ground connection (GND). ARC digital ports are labelled D0 through D23.

Output: When a digital port is set to True, it outputs +3.3V. When set to False, it outputs GND. This makes digital ports ideal for controlling devices that only need on/off control.

Input: When reading a digital port, ARC reports True if the voltage is above GND (and below +5V).
A short to GND is read as False.

Standard devices connected to digital ports include switches, buttons, servos, ultrasonic distance sensors, LEDs, and
simple logic-level peripherals.

ADC (Analog Input)

ADC stands for **Analog-to-Digital Converter**. These ports are **input-only** and measure varying voltages rather than simple on/off states. In ARC, ADC ports are labelled ADC0 through ADC7.

Reading relative voltage: ADC values can be returned as numeric ranges. In 8-bit mode, values range
from 0 to 255. In 12-bit mode, values range from 0 to 4095. These numbers represent a voltage range of 0–5V.
For example (8-bit): 0 = 0V, 127 ≈ 2.5V, 255 = 5V.

Reading absolute voltage: ARC can also return the actual measured voltage value in volts, making it easy
to monitor sensor output without manual calculations.

Typical analog devices include Sharp GP2 distance sensors, pressure sensors, light sensors, sound sensors, color sensors,
potentiometers, and general-purpose voltage monitoring.

Serial (TX Only)

Most digital ports on the EZ-B can transmit serial data. Serial communication sends data one bit at a time,
allowing more complex information to be transmitted than simple on/off signals. Some EZB models also include dedicated
high-speed UART ports for buffered communication.

Both the transmitting and receiving devices must be configured to the same baud rate (communication speed).
Standard baud rates include 300, 4800, 9600, 19200, 38400, 57600, and 115200 bits per second (bps).

Serial output is commonly used to control devices such as LCDs and motor or servo controllers, and to

communicate with microcontrollers like Arduino boards or robots such as the iRobot Roomba.

UART Serial (Bi-Directional)

UART ports provide bidirectional, accurate serial communication, enabling data to be sent and received. These ports connect
to TTL-level serial devices and include a 5,000-byte input buffer for incoming data. UART communication is handled using
the UARTInit(), UARTWrite(), UARTRead(), and UARTAvailable() commands in ARC.

UART baud rates can be set to any integer value between 1 and 3,750,000 bps, making them suitable for both slow sensors
and high-speed data devices.

I2C

I2C, also known as the **Two-Wire Interface**, uses two shared signal lines: Serial Data Line (SDA) and Serial
Clock Line (SCL). Multiple devices can be connected to the same I2C bus, and each device has a unique address.

I2C is commonly used for compact, low-pin-count devices such as LCD screens, I2C-enabled servos, BlinkM multicolor LEDs,
sensors, and other innovative peripherals.

Click here for more information on understanding I2C addressing.

COM Ports

A COM port is simply an I/O interface that enables the connection of a serial device to a computer. You may also hear COM ports referred to as serial or UART ports. Most modern computers are not equipped with COM ports, but many serial port devices are still in use that uses the interface. Lab instruments, medical equipment, and EZB Robot Controllers often use serial/UART connections.

So what are COM ports? They are asynchronous interfaces that transmit one bit of data simultaneously when connected to a serial device. The COM designation is due to their use as communication ports on IBM-compatible computers. In traditional personal computers, COM1 and COM2 were often used to connect a serial port device such as a modem or mouse. Here is what a DB9 COM port looks like.

What is a Serial/UART Device?

Serial devices, also known as UART, are electronic equipment characterized by how they transmit data. Information is sent and received one bit at a time, providing a simple and reliable method of communication.

What is a serial port on a computer?

Serial ports are not standard equipment on most newly manufactured desktop and laptop computers. The RS232 serial ports seen in the past are usually replaced with one or more USB interfaces or included with the USB driver of an Arduino. This does not mean you cannot connect a serial port device to a new computer. You can use a USB-to-serial adapter to provide single or multiple COM ports to machines without installed serial interfaces. Many solutions are available, and most are inexpensive and easy to implement. You will also notice that an Arduino will create a COM (serial) port when it is connected to the USB of a PC. That COM port for the Arduino is what you would use to connect ARC to it when the EZB firmware has been loaded.

The UART USB adapters or EZB Arduino work by installing a device driver on a Windows computer. The driver program enables the computer to use the USB to RS232 adapter to replace a serial interface. This is done by creating a virtual COM port that performs identically to a physical port on a machine's motherboard.

Types of serial ports

Serial ports come in three standard configurations. They are usually male connectors with either a nine or 25-pin connector or a pinout UART header for DIY use. The accepted manner of naming ports on personal computers labels the serial interfaces as COM1 and COM2. Each pin is used for a particular function, such as sending and receiving data or sending a terminal-ready message. Essentially, any interface compliant with the RS232 protocol is a serial port, including EZBs that have onboard UART.

EZBs That Use Serial Ports

Many EZB devices use the UART/Serial protocol. You will find serial connectivity built into devices like Arduino and the EZ-Robot EZ-B v4, to name just a few. Click here to view a list of EZBs and their connection type to see which are USB because that will mean COM port.

UART

UART stands for Universal Asynchronous Receiver/Transmitter. It is a hardware device that translates data between a computer's serial ports and the EZB. UART processing transmits a byte of data serially bit by bit. The information is then converted back to a whole byte at the other end of the connection.

View COM Compatible EZBs

Introduction To Servo Motors

This tutorial will provide technical information about servo-motors and how they work. We made it easy to get a robot up and running. However, there are many fun and exciting things to learn about how the robot works. The more you know, the more you can get your robot to do!

Table of Contents

- How To Move Servos In ARC
- What is a servo motor?
- Servo vs. PWM
- How does a servo motor work?
- Types of servo motors
- Controlling a servo motor
- How Does Continuous Rotation Servo Work?
- How is torque measured?
- PWM (hobby servos) vs. Serial (smart) Servos
- ARC Servo Robot Skills

How To Move Servos in ARC

We provided this support document for those who wish to learn how a servo works; however, press this button to learn how to move servos in ARC.

Learn how to move servos in ARC

What is a servo motor?

A Servo motor (or servo) is a rotary actuator that allows precise angular position, velocity, and acceleration. Servos are found in many places: toys, home electronics, cars, and airplanes. If you have a radio-controlled model car, airplane, or helicopter, you are using at least a few servos. Servos also appear behind the scenes in devices we use every day. Electronic devices such as DVD and Blu-ray discPWMTM players use servos to extend or retract the disc trays.

Synthiam uses servos that manage the movement of joints, pan & tilt, and continuous rotational movement. For PWM Servos, the EZB sends an electrical signal that tells the servo what position to reach and how quickly to get there. Smart serial servos receive a protocol command by the EZB and the respective robot skill. Servos come in a variety of shapes and sizes for different applications. You may want a large, powerful one for moving the arm of a giant robot or a tiny one to make a robot's eyebrows go up and down. By linking many of these servos together, you can very easily create robots that perform complex real-world operations.

Hobby PWM Servo vs. PWM

PWM stands for Pulse Width Modulation. PWM is the process of turning ON and OFF digital

voltage quickly to simulate a range of voltage. For example... If the digital output pin of a 5v micro is 2.5v, then the PWM is set for a 50% duty cycle. If the digital output of a 3.3v pin is 1.65v, then the PWM is set for a 50% duty cycle. The microcontroller turns the digital 3.3v pin on and off quickly, producing a simulated lower voltage. You can use PWM to vary the brightness of an LED, for example.

A hobby servo uses PWM as well. The "frame" of a servo PWM signal is 20ms. This means the signal frame repeats every 20ms. The signal is only 2ms, where the rest of the frame is empty (low). Many controllers, such as Arduino libraries, do not maintain the 20ms specification defined for servos. Because of this, challenges have been introduced to servo manufacturers when decoding incoming PWM signals. This has caused the need for servos to be "Smarter" by adapting to the unusual PWM transmitted by poorly written libraries that do not adhere to the servo PWM Standard. The EZ-B does adhere to servo PWM standards.

How does a servo motor work?

The simplicity of a servo is among the features that make them so reliable. The heart of a servo is a small direct current (DC) motor, similar to what you might find in an inexpensive toy. These motors run on electricity from a battery and spin at high **RPM** (rotations per minute) but put out very low **torque** (a twisting force used to do work — you apply torque when you open a jar). An arrangement of gears takes the high speed of the motor and slows it down while at the same time increasing the torque. (Basic law of physics: work = force x distance.) A tiny electric motor does not have much torque, but it can spin fast (small force, considerable distance). The gear design inside the servo case converts the output to a much slower rotation speed but with more torque (significant force, little distance). The amount of actual work is the same, just more practical. Gears in an inexpensive servo motor are generally made of plastic to keep them lighter and less costly. On a servo designed to provide more torque for heavier work, the gears are made of metal (such as with Synthiam Servos) and are harder to damage.

You apply power from a battery with a small DC motor, and the motor spins. However, unlike a simple DC motor, a servo's spinning motor shaft is slowed way down with gears. A positional sensor on the final gear is connected to a small circuit board. The sensor tells this circuit board how far the servo output shaft has rotated. The electronic input signal from the computer or the radio in a remote-controlled vehicle also feeds into that circuit board. The electronics on the circuit board decode the signals to determine how far the user wants the servo to rotate. It then compares the desired position to the actual position and decides which direction to rotate the shaft to get to the desired position.

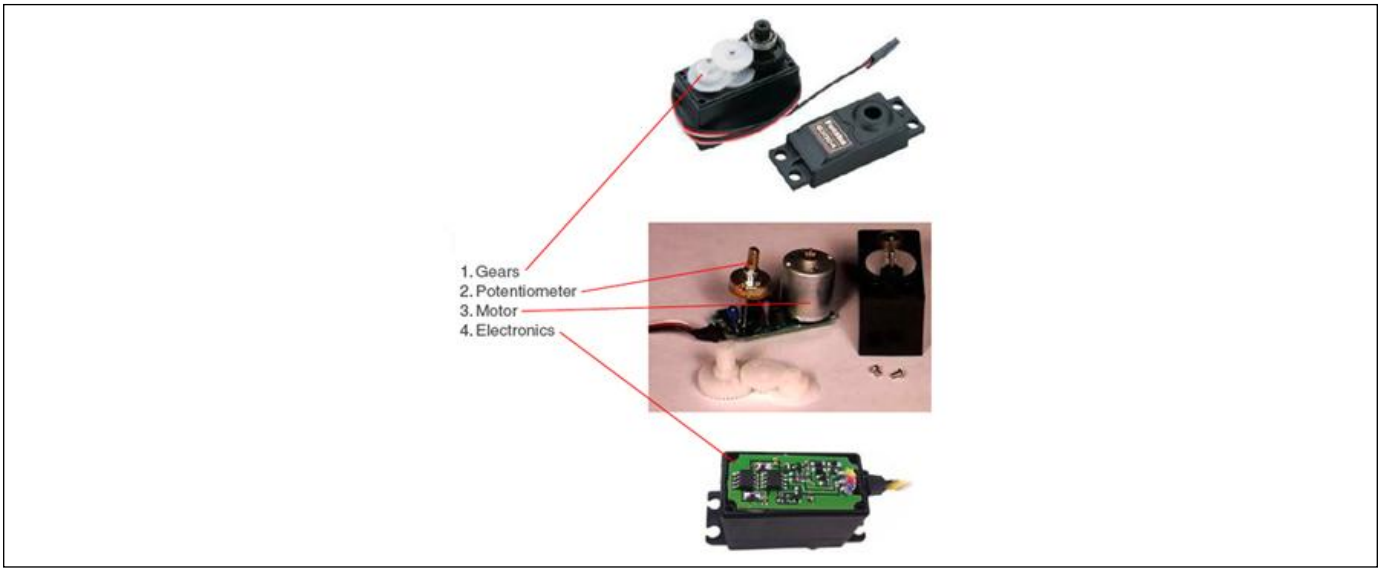


Figure 5. The circuit board and DC motor are in a high-power servo. Did you notice how few parts are on the circuit board? Servos have evolved to a very efficient design over many years.

Types of servo motors

Servos come in many sizes and three basic types: positional rotation, continuous rotation, and linear. Regardless of the type of servo between PWM Hobby and Smart serial, the servo form factor can be in various configurations.

- **Positional rotation servo**: This is the most common type of servo motor. The output shaft rotates in about half of a circle or 180 degrees. It has physical stops placed in the gear mechanism to prevent turning beyond these limits to protect the rotational sensor. These common servos are found in Synthiam 's arms, legs, limbs, etc. For example, the JD or Six robots use these servos. Some servos, such as winch servos, will rotate their output shaft in multiple rotations (i.e., 270 degrees). The same signal (pulse width) is used for any servo regardless of the output shaft rotation. This means a servo of 270 degrees would have less resolution than a servo with a 180-degree range/
- **Continuous rotation servo**: This is quite similar to the common positional rotation servo motor, except it can turn in either direction indefinitely without any positioning ability. This means the servo does not know its position because it operates as a motor and spins continuously in either direction. There is no way to specify the position of a continuous rotation servo because they lack the hardware to do so. The control signal, rather than setting the static position of the servo, is interpreted as the direction and speed of rotation. The range of possible commands causes the servo to rotate clockwise or counterclockwise as desired, at varying speeds, depending on the command signal. If you mounted one on a robot, you might use a servo of this type on a radar dish. Or you could use one as a drive motor on a mobile robot. The Synthiam AdventureBot uses two continuous rotation servos with wheels. The center (90 degrees) is the center of a continuous rotation's STOP position. The further you move away from 90 degrees in either direction controls the speed of the continuous rotation servo in that direction.
- **Linear servo**: This is also like the positional rotation servo motor described above, but with additional gears (usually a **rack and pinion** mechanism) to change the output from circular to back-and-forth. These servos are not easy to find, but you can sometimes find them at hobby stores, used as actuators in larger model airplanes.

Controlling a PWM servo motor

A standard servo is what you usually find in R/C Hobby Toys. They are high-precision devices that can rotate a shaft up to 180 degrees. With the EZB and a Standard Servo, you can easily configure how many degrees to rotate the output shaft.

ARC takes care of the electrical communication with the servo for you, whether PWM or Smart serial. You can use standard servos for the head or arms of your robot.

However, here is some technical information on how hobby PWM servos work. The servo is controlled using pulse control. The control pulse is a positive voltage with a length of 1 to 2 ms, which determines the angle of the shaft. The control pulse is repeated every 18-25 ms.

TO ACCOMMODATE ALL SERVO TYPES, the EZ-B has timing below 1ms and above 2ms. Some servos do not fall within the specifications and require unusual timing. When testing

with your servo, make sure you recognize the max and min values and set them in the robot skill settings. If a servo attempts to move further than its maximum position, it may be damaged. Additionally, if a servo is rotated too far, it will consume a high current, and the EZ-B may reset.

Here are the timings for the EZ-B...

Position 1 on v4, Position 1 on v3

Position 90 on v4, Position 50 on v3

Position 180 on v4, Position 100 on v3

For example, the EZ-Robot EZ-B v4 has high accuracy, which results in 180 servo positions. The EZ-B v4 can control 24 servos simultaneously while performing other user-specified tasks. If your servos draw more current than our specification sheet defines, the EZ-B may run out of power and reboot itself. This is called a Brown-Out. To prevent brownout with many servos, provide alternate power. Check the manual on how to do that.

ARC handles the technical work for you. Merely specify the servo position, and voila!

How Does Continuous Rotation Servo Work?

As mentioned earlier in this lesson, a continuous rotation servo will spin the rotation shaft continuously in either direction. The continuous rotation servo will stop spinning when the servo receives a 90-degree position centered on a standard servo. The center (90 degrees) is the center of a continuous rotation's STOP position. The further you move away from 90 degrees in either direction controls the speed of the continuous rotation servo in that direction. The continuous rotation servo will have a POT (potentiometer) exposed, allowing you to fine-tune the STOP position with a small screwdriver. Click
here
to view a tutorial on calibrating a continuous rotation servo.

100 < - Turning left faster

99

98

97

96

95

94

93

92

91 < - Turning left slower

90 <- Stopped

89 <- Turning right slower

88

87

86

85

84

83

82

81

80 < - Turning right faster

How Is Torque Measured?

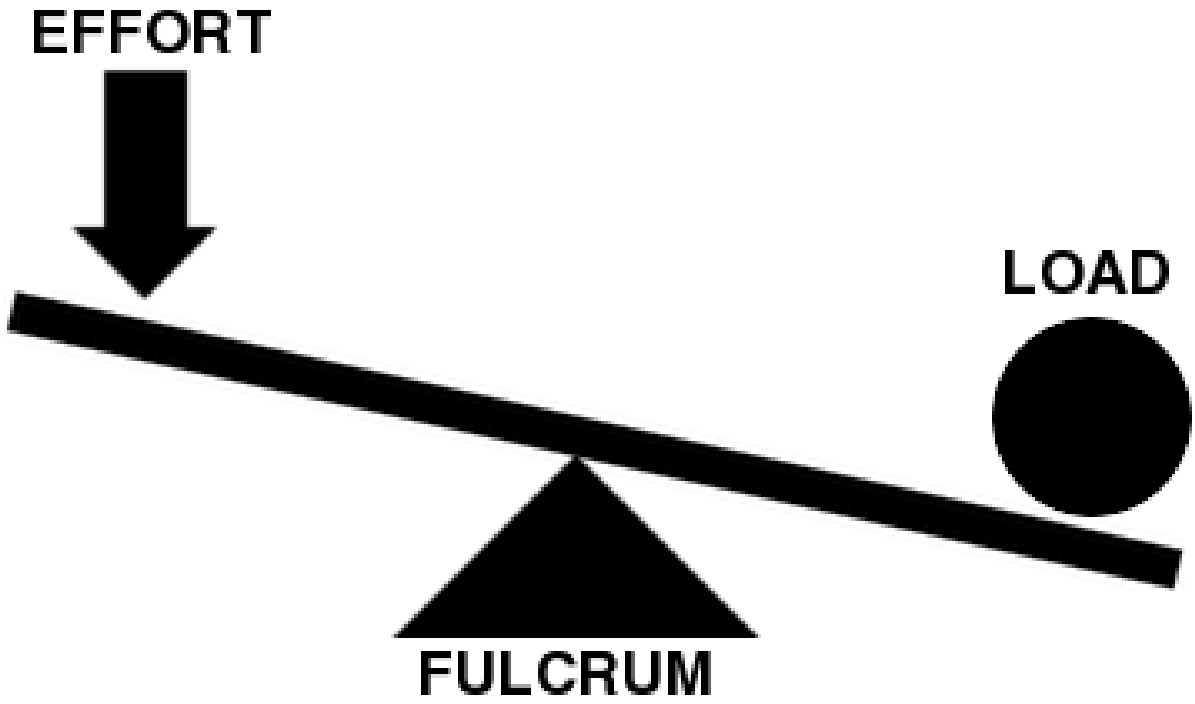
Now that we covered the basics of how-a-servo-works, the next question is torque. Torque is how much power the servo has. Different applications will require higher or lower torque. In most cases, you can get away with using a regular average torque servo. The average torque of a plastic servo is 2-3kg/cm @ 5 volts, and the Synthiam Heavy Duty servos are much more (consult the servo product information).

Something that is often overlooked is energy consumption. Consider this: there is no free energy. Torque equals energy and vice-versa. The Synthiam Heavy Duty servos will require more energy to move than cheaper plastic ones. Suppose too many high torque servos are connected to the EZ-B without a sufficient power source. In that case, it will "brownout" Browning-out means the voltage regulator could not keep up with the current draw, so the microchip rebooted itself due to low current. Do not worry; the Revolution Robots are powered by a strong LiPo battery and support up to 24 servos!

So what do the torque numbers mean? Let's speculate the torque value of a servo was 50 Ounces per Inch.

Well, if you had a servo arm that was one inch long on your servo, it would be able to produce 50 ounces of pull or push force at the end of the servo arm before stalling. What do

you think the force would be if you had a 1/2 inch servo arm? Yup, 100 ounces of force. How about a 2-inch arm, 25 ounces of force - easy, huh?



PWM (hobby servos) vs. Serial (Smart) Servos

The most common servos in robotics are three-wire PWM Hobby Style (power, GND, PWM signal). These servos have been around for decades and are the most cost-effective solution. The shortcoming of these servo types is they have only one direction of communication from EZB to the servo. This means the servo only knows its position when instructed to move. When these servos are first initialized, they will jump from whatever position is currently into the specified position. It will not transition slowly during initialization because it does not know the starting position before initialization. When robots use these servos, they will jump and sometimes cause damage preparation is not considered. The solution to avoid jumping is to manually move the robot joints into a position close to the initialization position so the servos don't jump. This is more effort than using Smart Servos, which will report their position to the software.

Smart serial servos use a communication protocol that allows the servos to transmit position data to the EZB. These smart servos have bi-directional communication to query the position when initialized and transition to the new position at specified speeds. This prevents jolting/jumping of joints that are experienced with hobby PWM servos.

When a robot is powered on, it is recommended to create an initialization script launched from the connection control. The initialization script will accommodate the type of servos being used. If the servos are PWM style, it is recommended to move the robot joints before executing the initialization script manually. With a smart serial servo, you can query the position. Therefore ARC will know the position, and your script can calmly move the servos into a pre-defined position.

ARC Servo Skills

ARC supports several servo robot skills.

[opt:skillcategory:servo]

Robot Topology

Let's break down the layers of a robot.
ARC runs on an x86 Windows-based PC, Laptop, or SBC (single board computer) to leverage the PC’s powerful architecture and hardware. The connectivity between ARC and sensors, servos, and peripherals is through a supported I/O controller. Also, the PC provides USB extensibility for additional peripherals, such as joysticks, cameras, audio devices, and more.