IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

# Bridging Hardware and Firmware: An End to End Approach to Dynamixell Servo

**[1]Name of 1st Author: Daiwik Selvakumar**
[1]Designation of 1st Author: Student
[1]Name of Department of 1st Author: Student
[1]Name of organization of 1st Author, City, Country: Indus International School Pune, India

*Abstract:* **:** Dynamixel servo motors have emerged as a robust solution for advanced robotics, offering integrated control, feedback, and customizable interfaces. This paper covers AC-to-DC power conversion, Zener diode regulation, and advanced circuitry to ensure reliable performance. Finally, the paper guides users through Wizard 2.0 coding to seamlessly integrate Dynamixel servos into diverse applications.

## THE DYNAMIXELL SERVO MOTOR

Dynamix cell servo motors are motors which work smartly. These motors are motors which have all the independent components like a motor, sensor, circuit board, etc into one motor. Since the components are combined and are working **dynamically**, it is called a dynamic cell servo motor.

A dynamic cell servo motor consists of:

1. **Encoders**
2. **Gear set**
3. **Motor Unit**
4. **Controller circuit**
5. **Network interface**

In this section of the paper I plan to cover all the above aspects of this motor, how it works, its parts, its steps, its real life applications, and its types of controls.

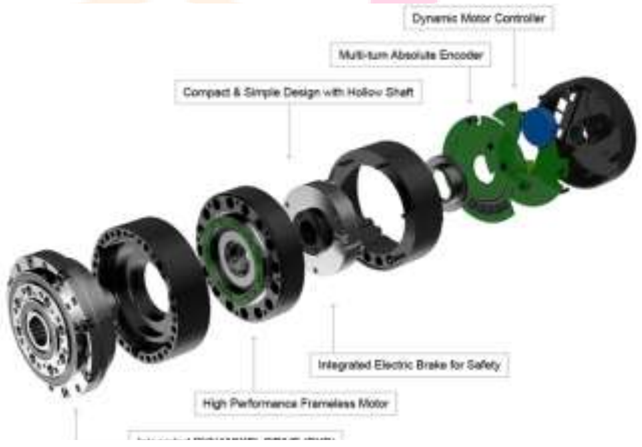

Dynamic Motor Controller
Multi-turn Absolute Encoder
Compact & Simple Design with Hollow Shaft
Integrated Electric Brake for Safety
High Performance Frameless Motor

*Figure 1.1 Structure of a Dynamixell motor*

**Encoders:**

Encoders are devices which measure the position and sometimes the speed of the output shaft of the motor. There are 2 types of encoders:

1. Absolute Encoders
2. Incremental Encoders

Absolute Encoders:

Absolute Encoders are encoders which are special for their:

1. Uniqueness: The Absolute encoders provide the exact data for the position or speed of the output shaft.
2. Power loss: Absolute encoders will remember the position or speed of the output shaft after a power loss also. This helps the encoder as when the power get backs on it does not need to reentry the data of the position it stopped at.
3. Data transmission: The data transmission of an absolute encoder is done through a binary code or a gray code to transmit information.
4. Applications: When it comes to application these encoders are very useful for applications which require a precise position of the output shaft.

Incremental Encoders:

Incremental encoders are encoders which are special for their:

1. Relative position: Incremental encoders produce signals to determine how much the shaft has moved but not its exact position. To know its exact position there needs to be a start from a known reference point.

2. Power loss: When an incremental encoder occurs through a power loss, the full process of the encoder will have to restart from a known a position it has already been to.
3. Pulses: Pulses are let out by this encoder. This is used to determine the movement of the output shaft using a control system.
4. Quadrature output: These encoders use two output's A and B, which are out of phase (Quadrature). This way the encoder can not only determine the amount but also the direction of rotation.

Comparison of Incremental and Absolute Encoders

| S.No. | Comparison | Description |
|---|---|---|
| 1 | Complexity and Cost | Absolute encoders are normally more unique and expensive than an incremental encoder as they must produce a unique code for each position |
| 2 | Precision | Absolute encoders are preferred for applications where safety and precision are the utmost priority. |
| 3 | Simplicity | Incremental encoders are simple and at a low cost compared to the absolute encoders. |

Resolution:

Resolution is the smallest increment of measurement or motion that can be detected or controlled by the system.
In the context of encoders:

- Absolute encoder resolution: The resolution of an absolute encoder is determined by the number of unique positions the encoder can differentiate. (Measured in bits)
- Incremental Encoder resolution: The resolution of an incremental encoder is determined by the number of pulses per revolution given out by the encoder.

**Gear Set:**
Dynamixell motors require gears to function. Gears are the muscle to the motor which act towards the decisions made by the controller circuit.
Dynamixell Servo motors mostly use planetary gears due to their high efficiency and compact size. The planetary gears consist of the main big ring which is the sun ring, and the other planet rings (The small rings) Which revolve around it.

This setup allows for a high torque output in a small space which is perfect for robotic applications where space is limited.

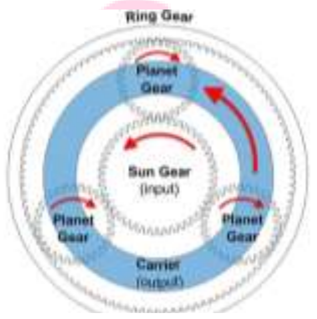


*Figure 1.2 Structure of a Gear*

**Motor Set:**
Motor Set in a dynamixell servo motor:
A Dynamixell Motor typically consists of a DC Motor which is coupled with the gear set to provide torque and speed. This torque and speed is normally decided by the encoder and is executed by the DC Motor. Here the Motor, Gear Set and encoder are all integrated into one set which works as the one functioning part of the Dynamixell servo motor. The type of DC motor is decided based on the factors of application required by the dynamixell motor to perform. This is how a particular series of dynamixell servo motors are picked like the AX series, MX series, XL series, XM series, etc.

Step:
Step comes from the word "Stepper Motors". The word "Step" can be determined by the smallest movement that a servo motor can control depending on the resolution of the encoder and the capabilities of the motor's controller. This makes sure the motor is at exact precision to what it does.
A common step size for a stepper motor is 1.8 degrees per step. This means that the motor moves 1.8 degrees with each command.

**Controller Circuit:**
Each dynamixell servo consist of a controller circuit and driver circuit. Both act together to handle the closed loop control, communications and status monitoring.

Closed loop control: Usually controls the position, velocity and torque of the motor.

Communication: Communicates to all the dynamixell servo parts using a microcontroller through a half-duplex line.

Status Monitoring: Makes sure that the dynamixell servo motor is in good condition and is provided the right amount of voltage, has a controlled temperature, and has a controlled load. This is basically the security system of the dynamixell motor.

All 3 of these closely work together to control each part of the dynamixell servo motor, make decisions based on the human commands inputted and monitor the safety of the motor to make sure all systems are in check.

Types of Controls in a controller Circuit:
There are various types of controls in a dynamixell motor. These controls are:

1. **Position Control:** Controlling the position of a motors output shaft to a specific angle (Can set limits to the position, like 180 degrees, 270 degrees, etc)
2. **Speed Control:** Controlling the rotational speed of the motor's output shaft.

3. **Torque Control:** Managing the amount of force, the motor applies.

**Network Interface:**
Dynamixell servo motors usually converse with the help of a RS485 and U2D2 Interface. This setup is normally used when there is a big amount of electronics being used. Using this interface will help communicate to other electronic devices easily from and to the dynamixell servo motor.

U2D2 and RS485 Components:
U2D2 Components:
1. TxD (Transmit Data): Sends data from U2D2 to other devices.
2. RxD (Receive Data): Receives data into U2D2 from other devices.
3. PWR (Power): Related to the power supply of the device.
4. RS485: A communication standard for long distances and noisy environments.
5. TTL (Transistor-Transistor Logic): A digital logic for serial communication.
6. UART (Universal Asynchronous Receiver/Transmitter): A protocol for serial communication without a clock signal.

RS485 Components:
1. GND- (Ground): The common electrical reference point.
2. VDD+ (Power Supply): The positive voltage supply for the transceiver.
3. Data (Data Line): Used for differential communication.
4. D- (Data Minus): The 'non-inverting' line for differential signaling.
5. D+ (Data Plus): The 'inverting' line for differential signaling.
6. DI and DE (Driver Input and Driver Enable): Control lines for the transceiver.

The Different Types of Communication Protocols:
1. I2C (Inter-Integrated Circuit): Allows connection between 2 ICs like a phone to a computer, etc.
2. UART (Universal Asynchronous Receiver Transmitter): Allows connection between a microcontroller and computer(GPS Modules)(TX-RX)
3. SPI (Serial Peripheral Interface): Allows short-distance connection between modules like microcontrollers and computers or between one or more ICs
4. USB (Universal Serial Bus): Allows connecting external devices like mouse, printer, etc to hardware.

Please Note: I2C, UART, SPI, and USB are circuit boards.

**How to Control A Dynamixell Motor Using Wizard 2.0:**
Wizard 2.0 is an app used to control mostly any dynamixell motor. It includes a lot of components, but below are the components which I did not understand in Wizard 2.0.

Components in Wizard 2.0:
1. ID: Unique Identifier for each dynamixell motor on each network.
2. Baud rate: Communication speed between the motor and controller. It is also measured in bps (Bits per second).
3. Return Time Delay: How long the motor waits before replying to the controller.
4. CW and CCW: Boundaries of how far a motor can turn clockwise (CW) and Anti-clockwise (CCW).
5. Status return Level: An option to decide how much time is needed to send back information.
6. Multi-turn offset and resolution divider: Option to let the motor spin continuously.
7. D, I, and P gain: Settings for motors brain deciding how to move. “D” is for how quickly the motor will react to changes, “I” is for making sure it doesn’t stop until you get to the right spot, and “P” is for how hard it pushes to get there.
8. Registered: Putting the command on wait. This way the motor knows you have asked for something but waits until you approve to do the command.
9. Punch: The amount of extra push given by the motor
10. Realtime tick: It’s like a clock inside the motor which helps coordinate if there are other motors involved.

Steps to program any dynamixell motor:
Step 1: Import library.
Step 2: Open and assign port for motor.
Step 3: Motor initialization (Enabling torque)
Step 4: Load the baud values in terms of degree or encoder position.
Step 5: Write the values.
Step 6: Close the port if there is no use again

Example code analyzation:
**Please note:** The code present here is a continuation. This is done to organize the code; hence the code is broken down.

**Importing:**

- from pypot.dynamixel import DxlIO
- import threading
- import time

In these 3 lines the process of importing is taking place. Here 'DxlIO' is being imported from 'pypot. dynamixel', threading is being imported, and time is being imported.

**Defining:**

- def eyes_function(dxl_io): #EYE
- servo_id_1 = 10
- velocity = 20
- dxl_io.set_moving_speed({servo_id_1: velocity})
- direction=1

These lines of code pick a port and define its name, speed, and direction. Once that is completed, a command is issued to begin doing the specific action.

**Initialization and Loading:**

```
while True:
target_angle1 = 90 #30
dxl_io.set_goal_position({servo_id_1: target_angle1})
current_position1 = dxl_io.get_present_position([servo_id_1])[0]
print("Motor 1 - Current position:", current_position1)
time.sleep(2)
```

This piece of code instructs the motor to continue doing the instructions below indefinitely unless instructed to stop. First, the motor's angle is defined. The motor is then directed to move. When moving, the location is examined, and a specific decision is taken to return to the proper angle. Finally, after that, the motor is instructed to hold awaiting additional directions.

```
if direction==1:
      print("DIRECTION 1")


      if int(current_position1) <= target_angle1:

        target_angle1 = 110#60
        while True:
          dxl_io.set_goal_position({servo_id_1: target_angle1})
          current_position1 = dxl_io.get_present_position([servo_id_1])[0]
          print("Motor 1 - Current position:", current_position1)
          if int(current_position1) <= target_angle1:

            direction=0
            time.sleep(1)
            break
```

This piece of code is largely comprised of if statements. The code begins with the condition: if the game's rule is to move forward (Direction=1), and if the motor has not achieved the desired angle, establish a new angle and advance towards it. If the motor reaches point B, reverse (Direction=0) and halt.

```
    else:

      if int(current_position1) >= target_angle1:

        target_angle1 = 70 #0
        while True:
          dxl_io.set_goal_position({servo_id_1: target_angle1})
          current_position1 = dxl_io.get_present_position([servo_id_1])[0]
          print("Motor 1 - Current position:", current_position1)
          if int(current_position1)-2 <= target_angle1:
            print("reached at 70")
            time.sleep(1)
            direction=1
            break
```

If the motor continues to move backward and has passed its previous angle, set a new angle and get there. Once at that point/angle, proceed onwards (Direction=1).

**Writing the values:**

```
def mouth_function(dxl_io):#MOUTH
    while True:
        file = open("/etc/Eagle_software/EagleCP/flag_file.txt", "r+")
        flag_value = int(file.read())

        if flag_value==1:

            servo_id_2 = 11
            velocity = 30
            dxl_io.set_moving_speed({servo_id_2: velocity})


            while True:
                target_angle2 =30 #MOUTH
                dxl_io.set_goal_position({servo_id_2: target_angle2})
                current_position2 = dxl_io.get_present_position([servo_id_2])[0]
                print("Motor2-Current_position:", current_position2)

                if int(current_position2) <= target_angle2:
                    target_angle2 =55
                    while True:
                        dxl_io.set_goal_position({servo_id_2: target_angle2})
                        current_position2 = dxl_io.get_present_position([servo_id_2])[0]
                        print("MOTOR2 second motion",current_position2)
                        if int(current_position2)+3>=target_angle2:
                            break
                break
        else:
            print("MOTOR 2 DISABELD")
            continue

    #print("mouth stop")
```

This code contains instructions for how the motor should move. The motor then comes into action. This will only happen if the user provides permission. Then the travelling speed, angle, and ID are defined numerically. After the numbers are entered, the code instructs the motor to do the tasks. A loop is constructed in which the code is repeated multiple times. Inside the loop, the motor has control over itself and will alter depending on its surroundings, speed, and angle. Then there's a quick check to see where the motor is, its speed, and angle. If the angle is not sufficient, the motor will adjust its target angle to a specific value (55); otherwise, the motor will continue the loop in the code to see where it takes itself.

**Closing:**

```
serial_port = "/dev/ttyUSB1"
baudrate = 1000000
dxl_io = DxlIO(serial_port)
dxl_io.enable_torque([10])
eyes_thread = threading.Thread(target=eyes_function, args=(dxl_io,))
mouth_thread = threading.Thread(target=mouth_function, args=(dxl_io,))

eyes_thread.start()
mouth_thread.start()
eyes_thread.join()
mouth_thread.join()
```

In this code, the defining of the baud rate, ports, torque, etc is defined and then there is an end to the code after there is no use to the code.

**Real Life Application of the Dynamixell Servo Motor:**

There are plenty of real life applications to this bot, but the 3 real life applications which struck my mind the most are Humanoid and Social bots, Industrial automation, and Prosthetics and Bionics.

Humanoid and Social Bots:

The Dynamixell servo motor is commonly used in social bots in order to control and perform various tasks like moving the hands, legs and neck of the robot. Controlled by a microcontroller, these bots could revolutionize the world in the fields of education, construction, industrialization, and a lot more. The open fine tuning in these motors allows these bots to even do precision work, like the work of a surgeon, ensuring that there is no mistake in the outcome because of the fine control of the small servos.

Industrial Automation:
Industrial Automation is a field which requires a lot of effort and energy. This could be easily replaced by robotic arms for assembling, manufacturing, and a lot more. Gears controlled by the servo motor could produce at a large scale ensuring efficiency and even producing during the night. Because dynamixell network is so efficient, it would be easy to mix and match multiple servo stations along a production line without complicated wiring. A lot of companies in the world have already implemented this but there is a long way for production to fully be controlled with the help of these motors.

Prosthetics and Bionics:
- Precision Limb Control: There's a lot of potential for smaller, high-torque servos to handle finger or wrist-level articulation in prosthetics, providing real-time feedback.
- Biofeedback Loops: When combined with EMG sensors or neural interfaces, these intelligent servos can adjust movement based on muscle signals from the user.

## ZENER DIODES AND THE LM8705

Zener Diodes let the electricity flow in the opposite direction (Reverse Biasing), which helps reduce the voltage. The voltage in a Zener diode is approximate while in the LM8705, it is exactly what it was programmed to do.

**How to convert a 230V AC supply to a 5V DC supply:**
1. Convert the 230V AC supply to a 12V AC supply using a step-down transformer which works when the primary coil in the transformer has more rotations than the secondary coil.

2. Then convert the 12V AC supply to a 12V DC Supply using the bridge rectifier diagram which works with diodes (Zener and normal), capacitors, and resistors.

3. Convert the DC Supply to a Square wave by using a voltage regulator called an LM7805 which gives an output of +5V only no matter what the input is.

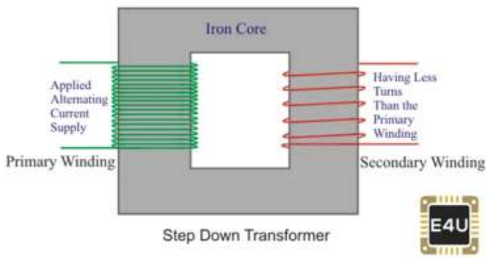
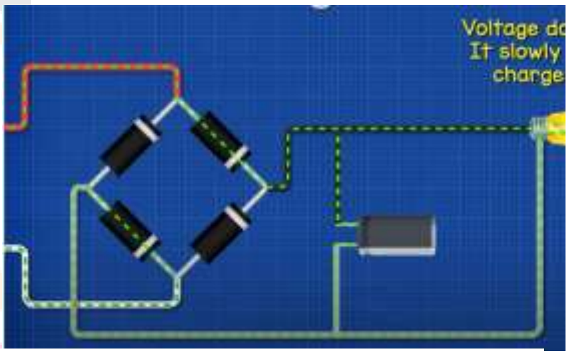

Iron Core
Applied Alternating Current Supply
Primary Winding
Having Less Turns Than the Primary Winding
Secondary Winding
Step Down Transformer
E4U

*Figure 1.3 Visualization of the Step Down Transformer*



The diagram shows the process of converting AC 230V to AC 12V, AC 12V to DC 12V, and DC 12V to DC 5V.

*Figure 1.4 The Bridge Rectifier Diagram*

PWM is a type of way to control the power control through PWM Signals. It's like moving a changing the angle of the tap. The angle of the tap here is the PWM signal, and the output of that PWM signal is the power/flow of water.

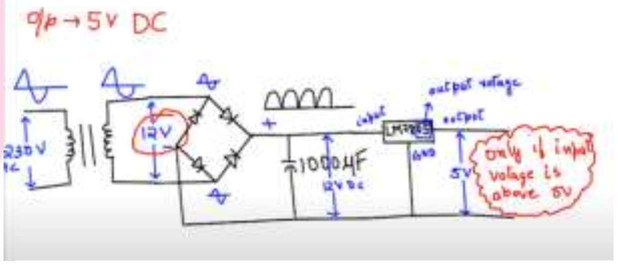


*Figure 1.5 Diagram Showcasing varied conversion of AC to DC power supply*

## MAJOR AND MINOR ELECTRONIC COMPONENTS IN AN ELECTRIC CIRCUIT

**Resistor:**
Types: Fixed, variable, and thermistor.
Normal Range: From 0.1 ohm to several megaohms.
Usage: To limit the flow of current and divide voltage in circuits. The by-product is heat.

**Capacitor:**
Types: Ceramic, electrolytic, tantalum, film, and supercapacitors.
Normal Range: From picofarads (pF) to farads (F).
Usage: To store and release electrical energy, filter signals, and block DC while allowing AC to pass.

**Inductor:**
Types: Air-core, iron-core, ferrite-core, and variable inductors.
Normal Range: From microhenries (μH) to henries (H).
Usage: To store energy in a magnetic field, filter signals, and in resonance circuits.

**Diode:**
Types: Signal diode, zener diode, light-emitting diode (LED), and photodiode.
Normal Range: Forward voltage drop typically 0.7V for silicon and 0.3V for germanium diodes.
Usage: To allow current to flow in one direction only, voltage regulation, and signal demodulation.

**Transistor:**
Types: Bipolar Junction Transistor (BJT), Field-Effect Transistor (FET), and Metal-Oxide-Semiconductor FET (MOSFET).
Normal Range: Various ranges based on type; for BJTs, current gain (hFE) can range from 20 to over 1000.
Usage: As a switch or an amplifier in electronic circuits.

**MOSFET:**
Types: N-channel and P-channel.
Normal Range: Threshold voltage typically ranges from 0.7V to 3V.
Usage: For switching and amplifying signals, in power applications due to high efficiency and fast switching.

**Relay Module:**
Types: Electromechanical and solid-state relays.
Normal Range: Coil voltage typically ranges from 5V to 24V for common modules.
Usage: To control a high-power circuit with a low-power signal, often used for isolation purposes.

**BJT (Bipolar Junction Transistor):**
Types: NPN and PNP.
Normal Range: Varies with design; collector current can range from mA to tens of A.
Usage: In switching and amplification, often found in audio and RF applications.

**Clippers:**
Types: Series and shunt, positive and negative.
Normal Range: Depends on the clipping level set by the diodes or other components used.
Usage: To clip or cut off parts of signal voltages beyond a certain level, used in waveform shaping.

**Clampers:**
Types: Positive and negative.
Normal Range: Depends on the reference voltage used for clamping.
Usage: To shift the level of a waveform without distorting the waveform's shape.

## BIBLOGRAPHY AND CITATIONS

- "DYNAMIXEL-Y ." *High Performance Premium Robot Actuator*, ROBOTIS, www.dynamixel.com/dxl_y.php. Accessed 1 Jan. 2025.
- "What Is a Planetary Gear?" *What Is a Planetary Gear?*, Regal Rexnord Corporation, 1 Jan. 2024, www.regalrexnord.com/regal-rexnord-insights/what-is-a-planetary-gear.
- "Paraphrasing Tool (AD-Free and No Sign-up Required) - Quillbot AI." *Quillbot*, 1 Jan. 2024, quillbot.com/paraphrasing-tool.
- AI, Open. "Chatgpt." *Chat GPT*, ChatGPT, 1 Jan. 2025, chatgpt.com/.
- "DYNAMIXEL." *Manuals*, ROBOTIS, 1 Nov. 2024, emanual.robotis.com/docs/en/dxl/.
- ROBOTIS. "TURTLEBOT3." *Manual*, 1 Dec. 2024, emanual.robotis.com/docs/en/platform/.
- Marek VACEK, Jaroslava ŽILKOVÁ, Marek PÁSTOR. "REGULATION OF DYNAMIXEL ACTUATORS IN ROBOT MANIPULATOR MOVEMENT." Technical University of Košice, 1 Dec. 2014.
- Ramadhani, Nur, et al. "Implementation of PID control for angular position control of Dynamixel servo motor." *Control Systems and Optimization Letters*, vol. 2, no. 1, 3 Jan. 2024, pp. 8–14, https://doi.org/10.59247/csol.v2i1.40.
- Smith, Andrew, and Jamil Jivraj. "Analysis of Robotis Dynamixel Ax-12+ Actuator Latencies (Publication by James Andrew Smith and Jamil Jivraj)." *Analysis of Robotis Dynamixel AX-12+ Actuator Latencies*, ROBOTIS, 22 Aug. 2022, forum.robotis.com/t/analysis-of-robotis-dynamixel-ax-12-actuator-latencies-publication-by-james-andrew-smith-and-jamil-jivraj/1148?utm.
- "Eagle Robot Lab." *Eagle Robot Lab*, 19 Jan. 2024, www.eaglerobotlab.com/.
- "Robotis E-Manual." *Manual*, 7 Jan. 2024, emanual.robotis.com/docs/en/software/dynamixel/dynamixel_wizard2/.
- TY - BOOK AU - Kuriakose, Alen AU - Solanki, Kharanshu AU - Tom, Meblu Sanand PY - 2021/04/25 SP - T1 - An Analysis of the Slayer Exciter Circuit VL - DO - 10.48550/arXiv.2104.12092 ER –

- Mindset, The Engineering. "Diodes Explained - The Basics How Diodes Work Working Principle Pn Junction." *YouTube*, YouTube, 12 Mar. 2021, www.youtube.com/watch?v=Fwj_d3uO5g8.
- God, Frozen, et al. "Capacitor Amplifying Voltage in Bridge Rectifier." *Electrical Engineering Stack Exchange*, Stack Exchange, 1 Dec. 1967, electronics.stackexchange.com/questions/642931/capacitor-amplifying-voltage-in-bridge-rectifier.
- Chemistry Tutor, The Organic. "220V AC to 12V DC Converter Power Supply Using Diodes, Capacitors, Resistors, & Transformers." *YouTube*, YouTube, 2021, www.youtube.com/watch?v=lKhmNanDkPY.