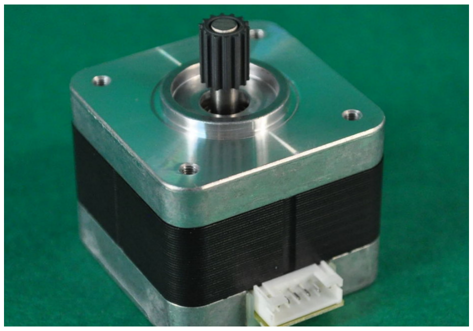
10. Motors with Binary Inputs

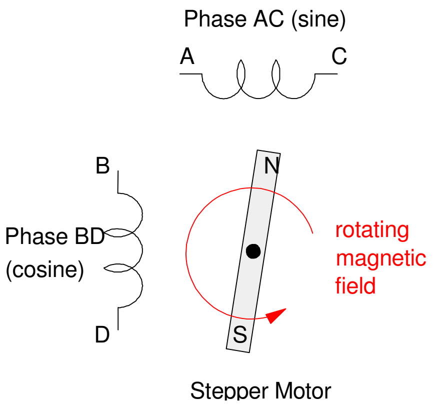
Introduction:

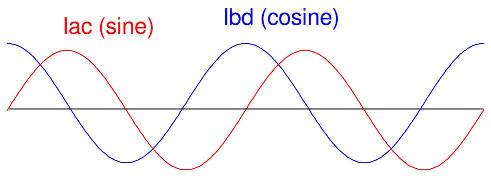
- Stepper Motors
- Servo-Motors (pulse-width)
- BLDC motors (pulse-width)
- Solenoids

Stepper Motors



Stepper motors are a common type of motor which interface well with microcontrollers. Stepper motors are actually 2-phase AC synchronous motors - meaning that if you apply a 2-phase sine wave (sine & cosine) to the inputs, you produce a rotating magnetic field, which causes the motor to spin smoothly.





The reason they're called *stepper motors* is if you approximate a sine wave with a square wave, the motor steps. The way you approximate a sine wave determines the number of steps per rotation - with three common methods called

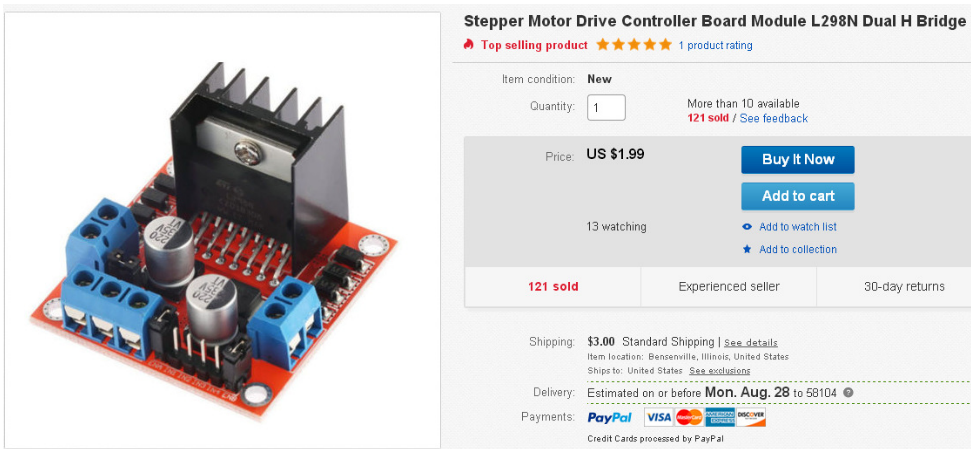
- Full-Stepping (four steps per cycle),
- Half-Stepping (eight steps per cycle), and
- Micro-Stepping (more than eight steps per cycle)

The total number of steps per rotation varies with the stepper motor. The ones we have in lab have 200 steps per rotation (for full-stepping) - meaning they actually have 50 electromagnetics around their perimeter. (50 sets of AC / BD coils around the circumference)

Stepper Motors: Hardware

Since current can flow both ways in each phase (since waves are positive and negative), an H-bridge is usually used to drive a stepper motor. A fairly inexpensive and pretty capable H-bridge is the L298N from ebay and Amazon, which is capable of

- 5V to 35V operation
- Up to 2A per phase
- Max power = 25W



Stepper Motor Drive Controller Board Module L298N Dual H Bridge

Top selling product ★★★★★ 1 product rating

Item condition: **New**

Quantity: 1 More than 10 available
121 sold / See feedback

Price: **US $1.99**

Buy It Now

Add to cart

13 watching

Add to watch list

Add to collection

121 sold | Experienced seller | 30-day returns

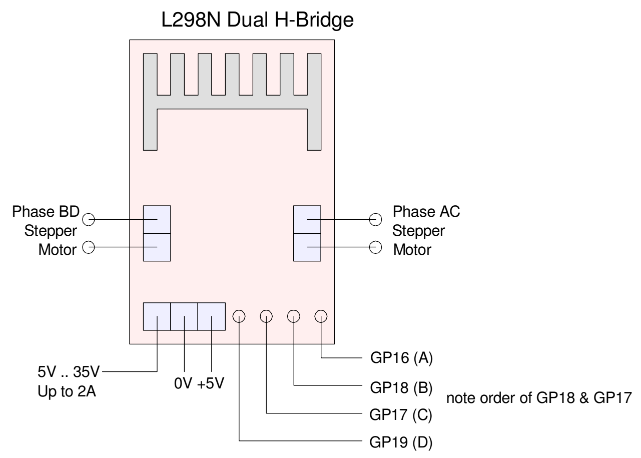
Shipping: $3.00 Standard Shipping | See details
Item location: Bensenville, Illinois, United States
Ships to: United States See exclusions

Delivery: Estimated on or before Mon. Aug. 28 to 58104

Payments: PayPal VISA
Credit Cards processed by PayPal

Dual H-Bridge Driver for a stepper motor.

Connections to your Pico board are:



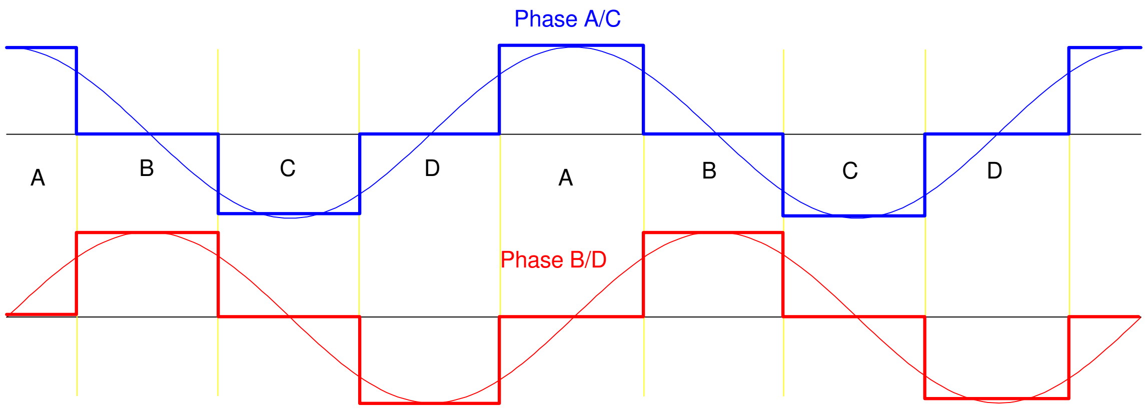
Four wires from the Pico needed to drive the stepper motor

Software - Full Stepping

With full-stepping, you approximate a 2-phase sine wave with square waves. This makes it easy for a Pico chip to drive the stepper motor. If {DCBA} is treated as a binary number the sequence to drive a stepper motor is

```
0001    A
0010    B
0100    C
1000    D
repeat
```

Swap the order and you step backwards



Full-Stepping approximates a 2-phase sine wave with four steps per cycle

In software, full-stepping can be implemented in code as follows:

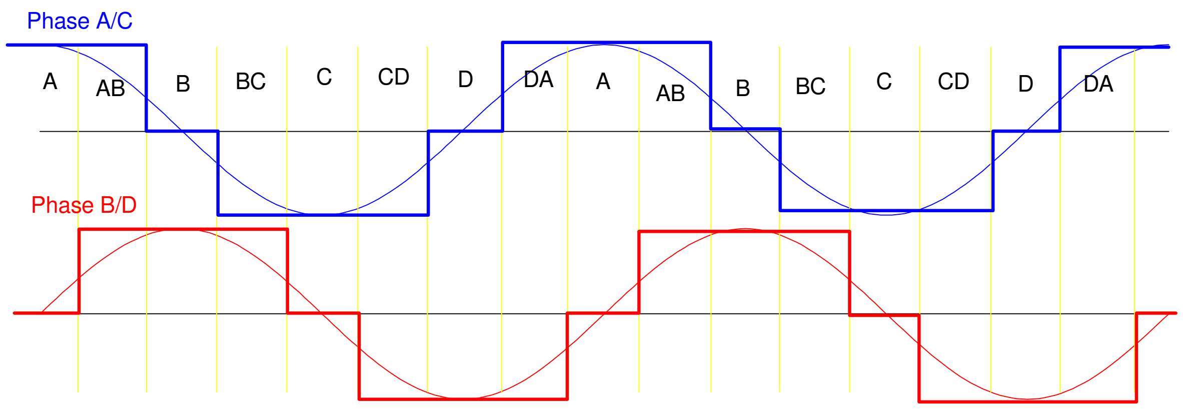
```
# Stepper Motor - Full Stepping

from time import sleep_ms
from machine import Pin

PA = Pin(16,Pin.OUT)
PB = Pin(17,Pin.OUT)
PC = Pin(18,Pin.OUT)
PD = Pin(19,Pin.OUT)

TABLE = [1, 2, 4, 8]

def Step(X):
    Y = TABLE[X % 4]
    PA.value( Y & 8 )
    PB.value( Y & 4 )
    PC.value( Y & 2 )
    PD.value( Y & 1 )

x = 0
for i in range(0,100):
    x += 1
    Step(x)
    sleep_ms(10)
```

Software - Half-Stepping

A slight change in software produces eight divisions per cycle - meaning eight steps per cycle. This is called *half-stepping*. If you treat the input as a 4-bit binary number {DCBA}, then the numbers written the output pins are:

```
0001    A       1
0011    AB      3
0010    B       2
0110    BC      6
0100    C       4
1100    CD      12
1000    D       8
1001    DA      9
repeat
```

For a stepper motor rated at 200 steps per rotation, you actually get 400 steps per rotation using half-stepping.



Approximation of sine / cosine wave using half stepping. Each cycle is broken down into eight steps.

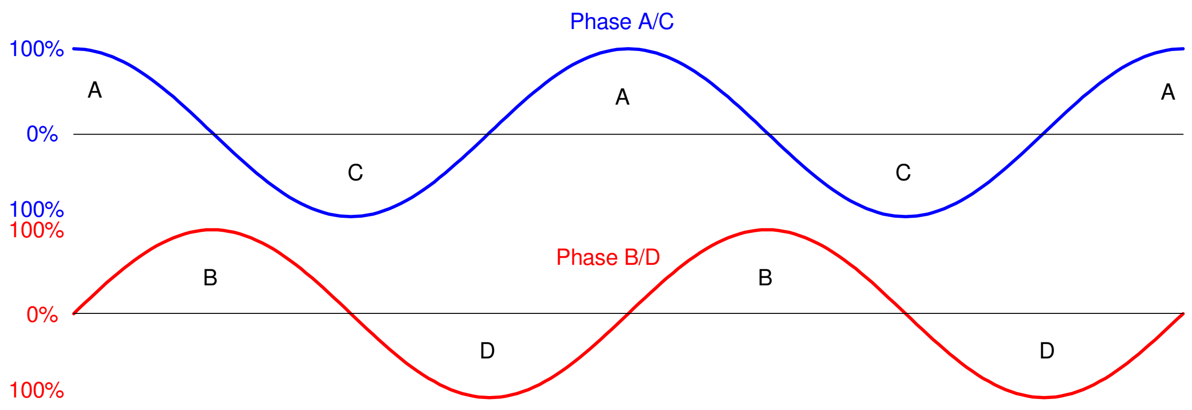
Code for half-stepping is almost the same, except that you have eight entries in your lookup table.

```
# Stepper Motor - Half Stepping

from time import sleep_ms
from machine import Pin

PA = Pin(16,Pin.OUT)
PB = Pin(17,Pin.OUT)
PC = Pin(18,Pin.OUT)
PD = Pin(19,Pin.OUT)

TABLE = [1, 3, 2, 6, 4, 12, 8, 9]

def Step(X):
    Y = TABLE[X % 8]
    PA.value( Y & 8 )
    PB.value( Y & 4 )
    PC.value( Y & 2 )
    PD.value( Y & 1 )

x = 0
for i in range(0,200):
    x += 1
    Step(x)
    sleep_ms(10)
```

Software - Micro-Stepping: 16 steps per cycle

A third option is to use PWM to approximate a sine and cosine wave. This is termed *micro-stepping*. The number of levels per cycle is arbitrary and can in theory be as fine as you want. For illustration purposes, we'll look at micro-stepping with 16 and 32 steps per cycle.

With PWM outputs, you can output anything from 0% to 100% duty cycle. To get negative voltages, you switch from phase A or B being energized (positive) to C or D being energized (negative):



PWM outputs allow you to generate any voltage from 0% to 100%

To speed up the computations, a look-up table is used for phase A (which is zero for the 2nd half of the sine wave). Phase B, C, and D can then be generated using the same look-up table and shifting where you read by 90 degrees, 180 degrees, and 270 degrees respectively.

The following code uses a look-up table with 16 entries - resulting in 800 steps per rotation

$$N = \left(\frac{\text{200 steps / rotation}}{\text{4 steps per cycle}}\right)(\text{16 entry table}) = 800\frac{\text{steps}}{\textit{rotation}}$$

```
from machine import Pin, PWM
from time import sleep_ms


PA = Pin(16,Pin.OUT)
PA = PWM(Pin(16))
PA.freq(100)
PA.duty_u16(0)

PB = Pin(17,Pin.OUT)
PB = PWM(Pin(17))

PB.freq(100)
PB.duty_u16(0)

PC = Pin(18,Pin.OUT)
PC = PWM(Pin(18))
PC.freq(100)
PC.duty_u16(0)

PD = Pin(19,Pin.OUT)
PD = PWM(Pin(19))
PD.freq(100)
PD.duty_u16(0)

TABLE16 = [0, 24874, 45962, 60052, 65000, 60052, 45962, 24874,  0,
 0,  0, 0, 0, 0, 0, 0]

def Step16(X):
    A = TABLE16[X % 16]
    PA.duty_u16(A)
    B = TABLE16[(X+4) % 16]
    PB.duty_u16(B)
    C = TABLE16[(X+8) % 16]
    PC.duty_u16(C)
    D = TABLE16[(X+12) % 16]
    PD.duty_u16(D)

x = 0
for i in range(0,800):
    x += 1
    Step16(x)
    sleep_ms(5)

PA.duty_u16(0)
PB.duty_u16(0)
PC.duty_u16(0)
PD.duty_u16(0)
```

Similarly, you could use a table with 32 entries, resulting 1600 steps per rotation:

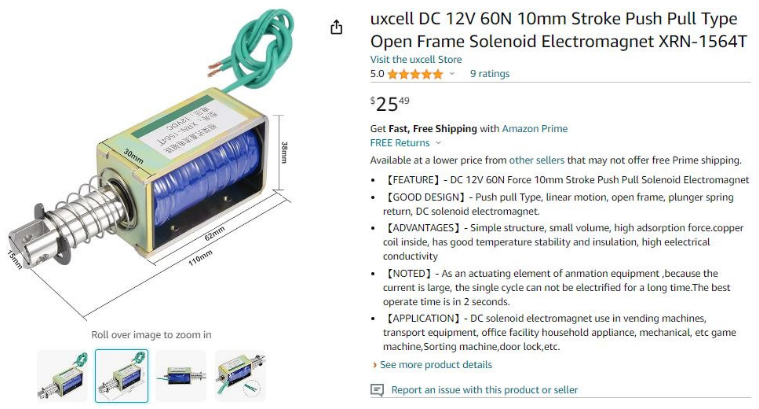
```
# Micro-Stepping;  32 steps per cycle
from machine import Pin, PWM
from time import sleep_ms

PA = Pin(16,Pin.OUT)
PA = PWM(Pin(16))
PA.freq(100)
PA.duty_u16(0)

PB = Pin(17,Pin.OUT)
PB = PWM(Pin(17))
PB.freq(100)
PB.duty_u16(0)

PC = Pin(18,Pin.OUT)
PC = PWM(Pin(18))
PC.freq(100)
PC.duty_u16(0)

PD = Pin(19,Pin.OUT)
PD = PWM(Pin(19))
PD.freq(100)
PD.duty_u16(0)

TABLE32 = [0, 12681, 24874, 36112, 45962, 54046, 60052, 63751, 
65000, 63751, 60052, 54046, 45962, 36112, 24874, 12681, 0, 0, 0, 0, 
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

def Step32(X):
    A = TABLE32[X % 32]
    PA.duty_u16(A)
    B = TABLE32[(X+8) % 32]
    PB.duty_u16(B)
    C = TABLE32[(X+16) % 32]
    PC.duty_u16(C)
    D = TABLE32[(X+24) % 32]
    PD.duty_u16(D)

x = 0
for i in range(0,1600):
    x += 1
    Step32(x)
    sleep_ms(5)

PA.duty_u16(0)
PB.duty_u16(0)
PC.duty_u16(0)
PD.duty_u16(0)
```

Solenoids

A solenoid is an electromagnet which can either pull or push a rod back and forth. Think of it as an electronic deadbolt:

- When de-energized, the deadbolt locks the door.
- When energized, the deadbolt is pulled back, allowing the door to open.

Since this is an of/off device, a simple binary output from the Pico can be used.



uxcell DC 12V 60N 10mm Stroke Push Pull Type Open Frame Solenoid Electromagnet XRN-1564T

Visit the uxcell Store

5.0 ★★★★★ 9 ratings

$25.49

Get **Fast, Free Shipping** with Amazon Prime

FREE Returns

Available at a lower price from other sellers that may not offer free Prime shipping.

- 【FEATURE】- DC 12V 60N Force 10mm Stroke Push Pull Solenoid Electromagnet
- 【GOOD DESIGN】- Push pull Type, linear motion, open frame, plunger spring return, DC solenoid electromagnet.
- 【ADVANTAGES】- Simple structure, small volume, high adsorption force.copper coil inside, has good temperature stability and insulation, high eelectrical conductivity
- 【NOTED】- As an actuating element of anmation equipment ,because the current is large, the single cycle can not be electrified for a long time.The best operate time is in 2 seconds.
- 【APPLICATION】- DC solenoid electromagnet use in vending machines, transport equipment, office facility household appliance, mechanical, etc game machine,Sorting machine,door lock,etc.

› See more product details

Report an issue with this product or seller

Sample Solenoid: Applying 12V to the leads draws 1A and applies 60N of force

Assume for example a uxcell 12V solenoid is to be driven by a Pi-Pico. The requirements are:

- V = 12V
- I = 1A @ 12V

Since a Pi-Pico can't output 12V or 1A directly, add a transistor switch (assume a ZTX1051A NPN transistor).

- Digikey Part: ZTX1051A
- Ic(max) = 4A
- DC Current Gain (min): 300 @ 1A, 2V
- Vce(sat) = 210mV @ 1000mA
- $0.68 (qty 100)

To saturate the transistor, you need

$$h_{fe} \cdot I_b > I_c$$

$$300 \cdot I_b > 1A$$

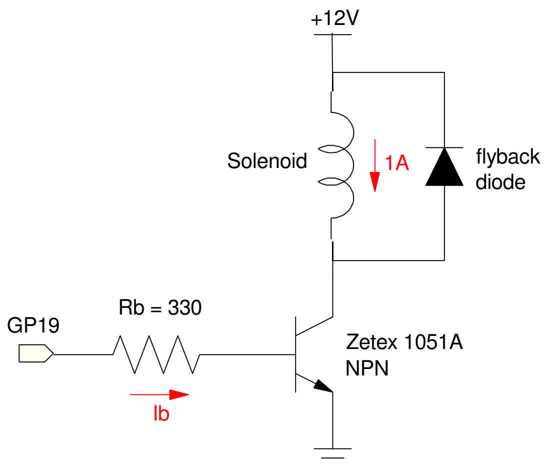
$$12mA > I_b > 3.33mA$$

(The 12mA is the maximum output from a Pi-Pico). Rb is then

$$R_b = \left(\frac{3.3V - 0.7V}{I_b}\right)$$

$$217\Omega < R_b < 780$$

Anything in this range should work. Let Rb = 330 Ohms. Note that solenoids are inductors - meaning you need to add a flyback diode to save the transistor when the inductor turns off.

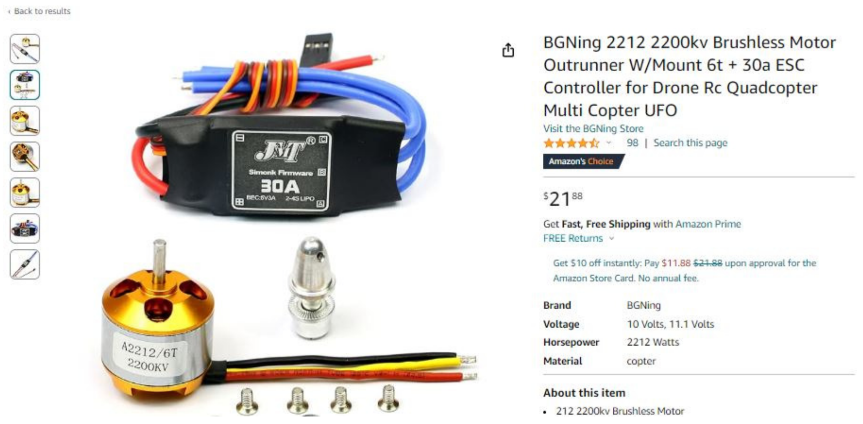


Solenoid Code:

A simple 1/0 on the output turns the solenoid on and off

```
# Turning a solenoid on and off
from machine import Pin
from time import sleep

GP19 = Pin(19,Pin.OUT)

while(1):
    print('Solenoid On')
    GP19.value(1)
    sleep(1)
    print('Solenoid Off')
    GP19.value(0)
    sleep(1)
```

Brushless DC Motors (BLDC - Quad-Copter Motors)



‹ Back to results

BGNing 2212 2200kv Brushless Motor Outrunner W/Mount 6t + 30a ESC Controller for Drone Rc Quadcopter Multi Copter UFO

Visit the BGNing Store

98 | Search this page

Amazon's Choice

$21.88

Get Fast, Free Shipping with Amazon Prime

FREE Returns

Get $10 off instantly: Pay $11.88 ~~$21.88~~ upon approval for the Amazon Store Card. No annual fee.

Brand	BGNing
Voltage	10 Volts, 11.1 Volts
Horsepower	2212 Watts
Material	copter

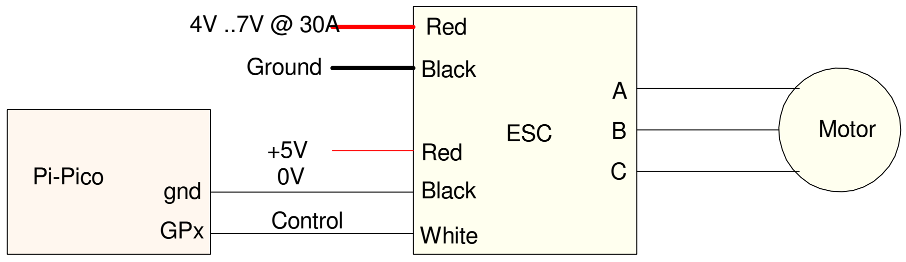
About this item

- 212 2200kv Brushless Motor

BLDC Motor (quad-copter motor)

A second class of motors with digital inputs use a PWM signal to control them, such as a BLDC motor. These motors use a 3-wire control input:

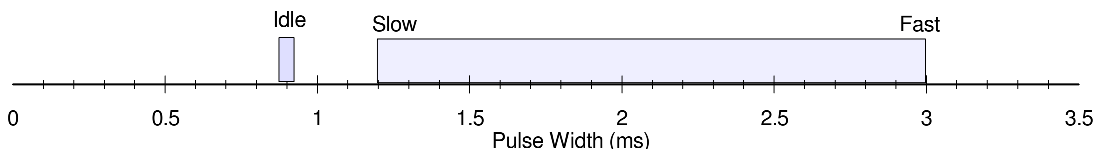
- Red: +3.3V
- Black: Ground
- White: Control Signal



Connections from a Pi-Pico to a BLDC motor.

The control input is a square wave:

- Frequency = 50Hz to 330Hz
- Stop (power on): 0.9ms pulse
- Slow: 1.2ms pulse
- Fast: 3.0ms pulse



The speed of the BLDC motor is set by the pulse width on the Control line

On power up. the ESC module wants to see a 0.9ms pulse on the control line. This is for safety: the blades of a quad-copter hurt when they hit you. If the ESC module sees the correct voltage and a 0.9ms pulse on the control line, it will then enter RUN mode, allowing you to vary the speed of the motor by adjusting the pulse width on the Control line.

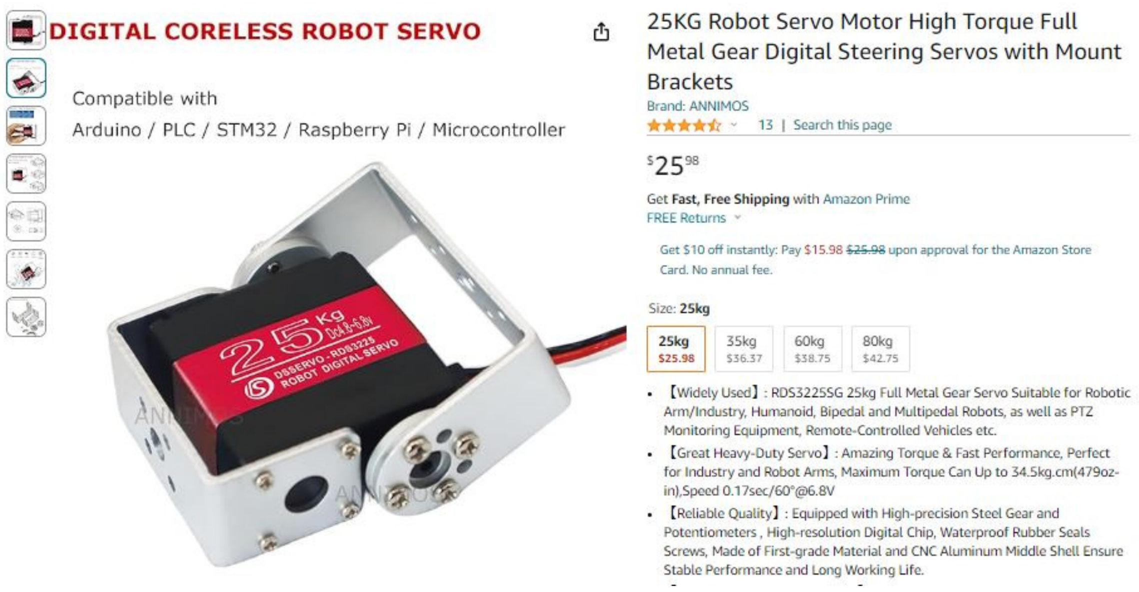
This is very easy to do with a Pi-Pico.

Sample Code

- Set the frequency to 50Hz (period = 20ms)
- Set the pulse width to 0.9ms until you press a button connected to GP15
- Once you press the button, switch from slow (1.2ms pulse) to fast (3.0ms pulse)

```
from machine import Pin
from time import sleep

# GP15 = push button
# GP16 = control input to BLDC

Button = Pin(15, Pin.IN, Pin.PULL_UP)

Control = Pin(16, Pin.OUT)
Control = PWM(Pin(16))
Control.freq(50)

Control.duty_ns(900_000)
while(Button.value() == 1):
    pass

while(1):
    print('slow')
    Control.duty_ns(1_200_000)
    sleep(1)
    print('fast')
    Control.duty_ns(3_000_000)
    sleep(1)
```

Digital Servo Motor


DIGITAL CORELESS ROBOT SERVO
Compatible with
Arduino / PLC / STM32 / Raspberry Pi / Microcontroller
25 Kg
DSSERVO - RDS3225
ROBOT DIGITAL SERVO

25KG Robot Servo Motor High Torque Full Metal Gear Digital Steering Servos with Mount Brackets

Brand: ANNIMOS

13 | Search this page

$25.98

Get Fast, Free Shipping with Amazon Prime

FREE Returns

Get $10 off instantly: Pay $15.98 ~~$25.98~~ upon approval for the Amazon Store Card. No annual fee.

Size: 25kg

25kg	35kg	60kg	80kg
$25.98	$36.37	$38.75	$42.75

- 【Widely Used】: RDS3225SG 25kg Full Metal Gear Servo Suitable for Robotic Arm/Industry, Humanoid, Bipedal and Multipedal Robots, as well as PTZ Monitoring Equipment, Remote-Controlled Vehicles etc.
- 【Great Heavy-Duty Servo】: Amazing Torque & Fast Performance, Perfect for Industry and Robot Arms, Maximum Torque Can Up to 34.5kg.cm(479oz-in),Speed 0.17sec/60°@6.8V
- 【Reliable Quality】: Equipped with High-precision Steel Gear and Potentiometers , High-resolution Digital Chip, Waterproof Rubber Seals Screws, Made of First-grade Material and CNC Aluminum Middle Shell Ensure Stable Performance and Long Working Life.

Typical Digital Servo Motor: Output is a 0 - 270 degree rotation

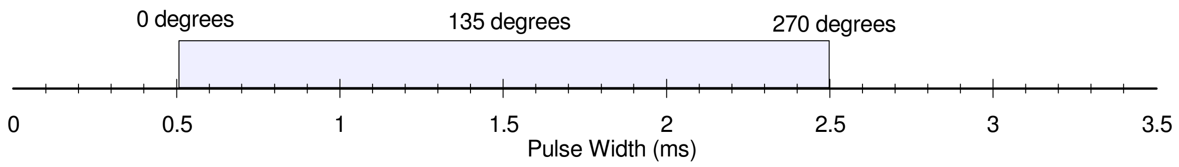
Another motor with a digital interface is a Digital Servo Motor. Think pan-and-tilt camera or a robotic arm for this motor.

The output of a Digital Servo Motor is an angle: the link can be set to 0-90 degrees, 0-180, or 0-270 degrees typically. Like the previous example, this is a 3-wire interface:

- Red: 5.0V to 6.8V, up to 3.0A (varies with the motor)
- Black: Ground
- White: Control Input

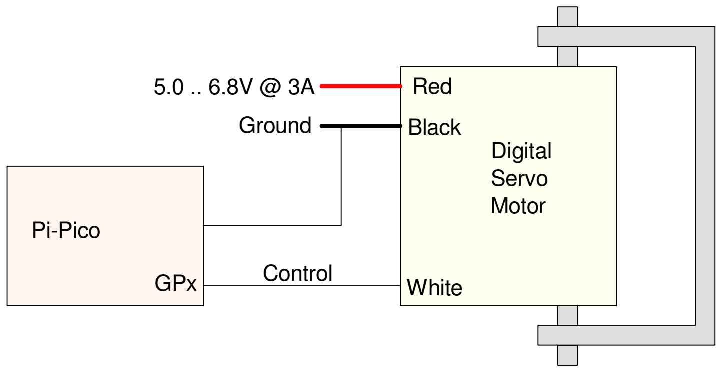
The control input sets the angle. For the motor given above, this is a square wave

- Frequency = 50 - 330Hz
- Pulse Width = 500 - 2500us
- Neutral Position = 1500us



The pulse width sets the angle of the motor

The hardware connection from a Pi-Pico to the motor is also similar to a BLDC motor:



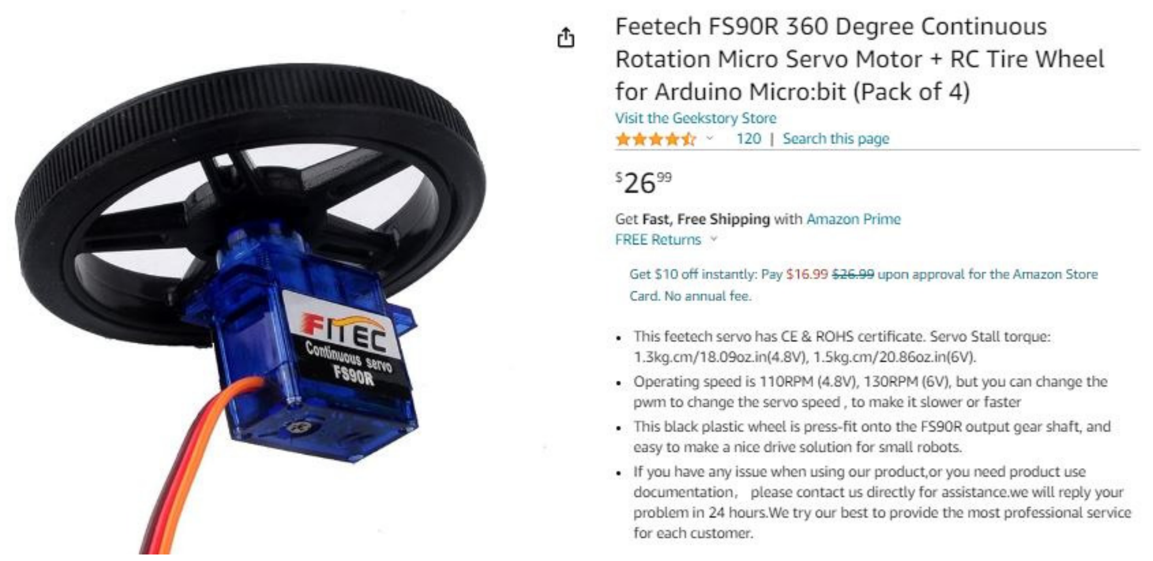
Connection from a Pi-Pico to a Digital Servo Motor.
The pulse width on the control line sets the angle.

The code is almost the same as before. As an example, the following rotates the arm

- Button 15: Increase the angle
- Button 14: Decrease the angle
- GP16: Control input

```
from machine import Pin
from time import sleep_ms

# GP14 = push button (decrease angle)
# GP15 = push button (increae angle)
# GP16 = control input to digital servo motor

Up = Pin(15, Pin.IN, Pin.PULL_UP)
Down = Pin(14. Pin.IN, Pin.PULL_UP)

Control = Pin(16, Pin.OUT)
Control = PWM(Pin(16))
Control.freq(50)

x = 1_500_000

while(1):
    if(Up.value() == 0):
        x += 1000
    if(Down.value() == 0):
        x -= 1000
    if(x < 500_000):
        x = 500_000
    if(x > 2_500_000):
        x = 2_500_000)
    Control.duty_ns(x)
    sleep_ms(1)
```

Continuous Rotation Servo Motor


Feetech FS90R 360 Degree Continuous Rotation Micro Servo Motor + RC Tire Wheel for Arduino Micro:bit (Pack of 4)
Visit the Geekstory Store
120 | Search this page

$26.99

Get **Fast, Free Shipping** with Amazon Prime
FREE Returns

Get $10 off instantly: Pay $16.99 ~~$26.99~~ upon approval for the Amazon Store Card. No annual fee.

- This feetech servo has CE & ROHS certificate. Servo Stall torque: 1.3kg.cm/18.09oz.in(4.8V), 1.5kg.cm/20.86oz.in(6V).
- Operating speed is 110RPM (4.8V), 130RPM (6V), but you can change the pwm to change the servo speed , to make it slower or faster
- This black plastic wheel is press-fit onto the FS90R output gear shaft, and easy to make a nice drive solution for small robots.
- If you have any issue when using our product,or you need product use documentation, please contact us directly for assistance.we will reply your problem in 24 hours.We try our best to provide the most professional service for each customer.

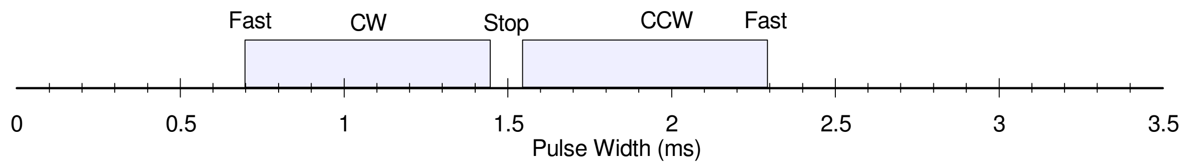
For speed control, a Continuous Rotation Servo Motor can be used

Finally, it you want continuous rotation, such as for driving the wheel of a car, a continuous rotation servo motor can be used. This again has a three-wire interface. For the example given here:

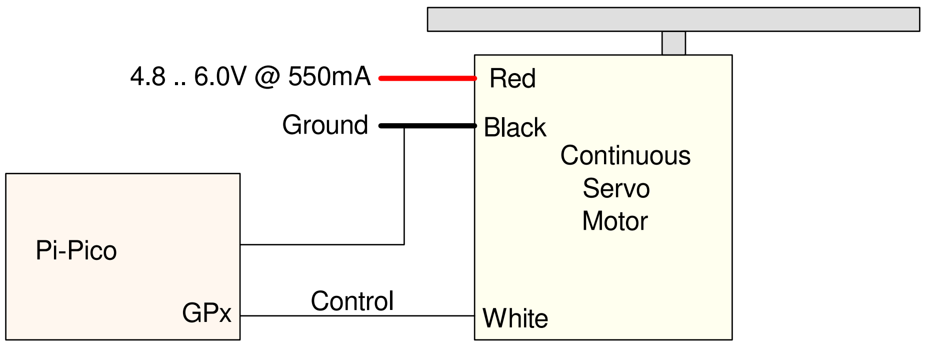
- Red = 4.8V - 6.0V
- Current: 100mA (no load) .550mA (stall)
- Pulse Width: 700 - 2300 us
- No-Load Speed: 110 rpm

The pulse width sets the speed:

- CW: 1500us - 700us
- CCW: 1500us - 2300us
- Stop: 1500us +/- 45us



Wiring to a Pi-Pico is similar to before:



Wiring from a Pi-Pico to a Continuous Servo Motor

Similarly, the code is almost the same as for a angle-control motor:

- Button 15: Increase the speed
- Button 14: Decrease the speed
- GP16: Control input

```
from machine import Pin
from time import sleep_ms

# GP14 = push button (decrease angle)
# GP15 = push button (increae angle)
# GP16 = control input to continuous servo motor

Up = Pin(15, Pin.IN, Pin.PULL_UP)
Down = Pin(14. Pin.IN, Pin.PULL_UP)

Control = Pin(16, Pin.OUT)
Control = PWM(Pin(16))
Control.freq(50)

x = 1_500_000

while(1):
    if(Up.value() == 0):
        x += 1000
    if(Down.value() == 0):
        x -= 1000
    if(x < 700_000):
        x = 700_000
    if(x > 2_300_000):
        x = 2_300_000)

    Control.duty_ns(x)

    sleep_ms(1)
```

Summary:

Digital motors are pretty easy to interface with a Pi-Pico:

- With a stepper motor, you mimic a 2-phase sine wave with four wires from the Pi-Pico
- With digital servo motors, you control the speed with a pulse width.

Note that these motors are low-power:

- The stepper motor draws 3A @ 5V, meaning 15W
- The digital servo motor draws 2A @ 5V, meaning 10W
- The continuous servo motor draws 550mA @ 5V, meaning 2.7W

Subtract losses in the motors and the power these can deliver is fairly small. If that's all you need, however, these are easy ways to interface motors to a Pi-Pico.

References

Pi-Pico and MicroPython

- https://github.com/geeekpi/pico_breakboard_kit
- https://micropython.org/download/RPI_PICO/
- https://learn.pimoroni.com/article/getting-started-with-pico
- https://www.w3schools.com/python/default.asp
- https://docs.micropython.org/en/latest/pyboard/tutorial/index.html
- https://docs.micropython.org/en/latest/library/index.html
- https://www.fredscave.com/02-about.html

Pi-Pico Breadboard Kit

- https://wiki.52pi.com/index.php?title=EP-0172

Other

- https://docs.sunfounder.com/projects/sensorkit-v2-pi/en/latest/
- https://electrocredible.com/raspberry-pi-pico-external-interrupts-button-micropython/
- https://peppe8o.com/adding-external-modules-to-micropython-with-raspberry-pi-pico/
- https://randomnerdtutorials.com/projects-raspberry-pi-pico/
- https://randomnerdtutorials.com/projects-esp32-esp8266-micropython/