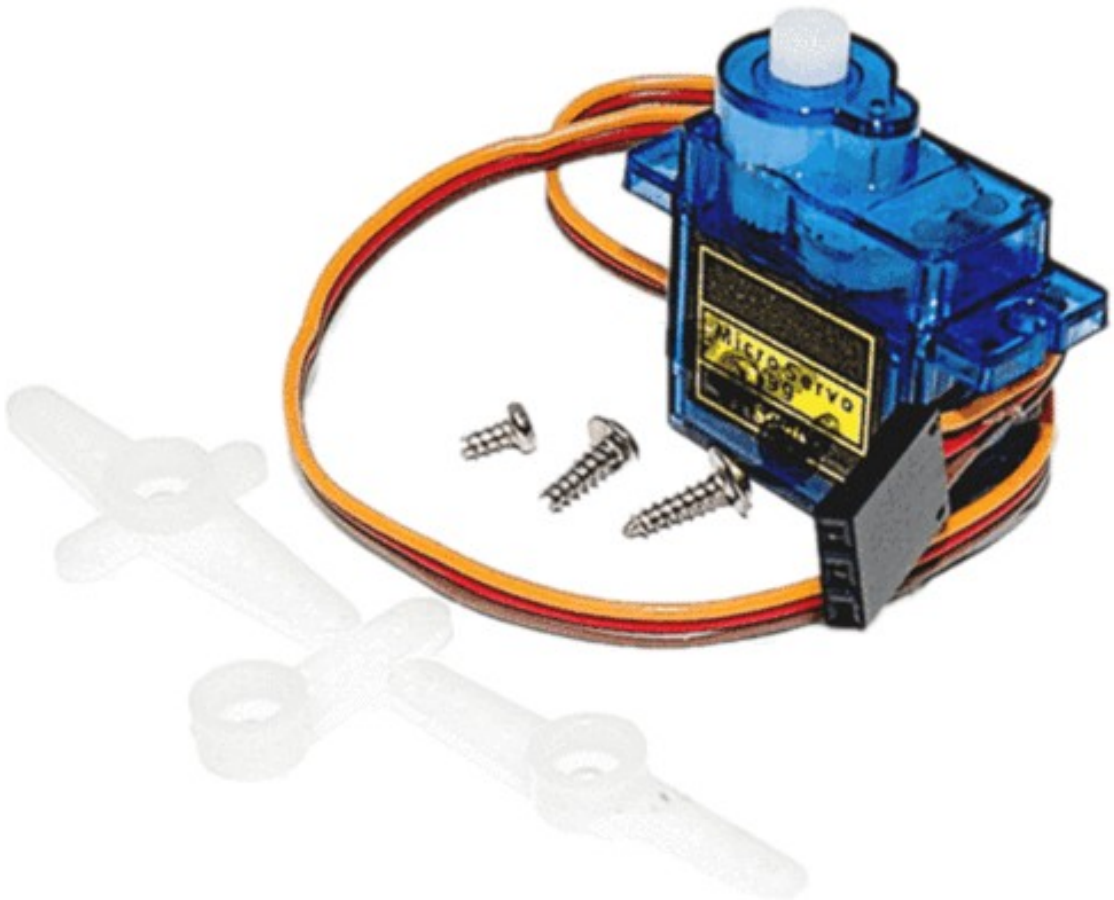
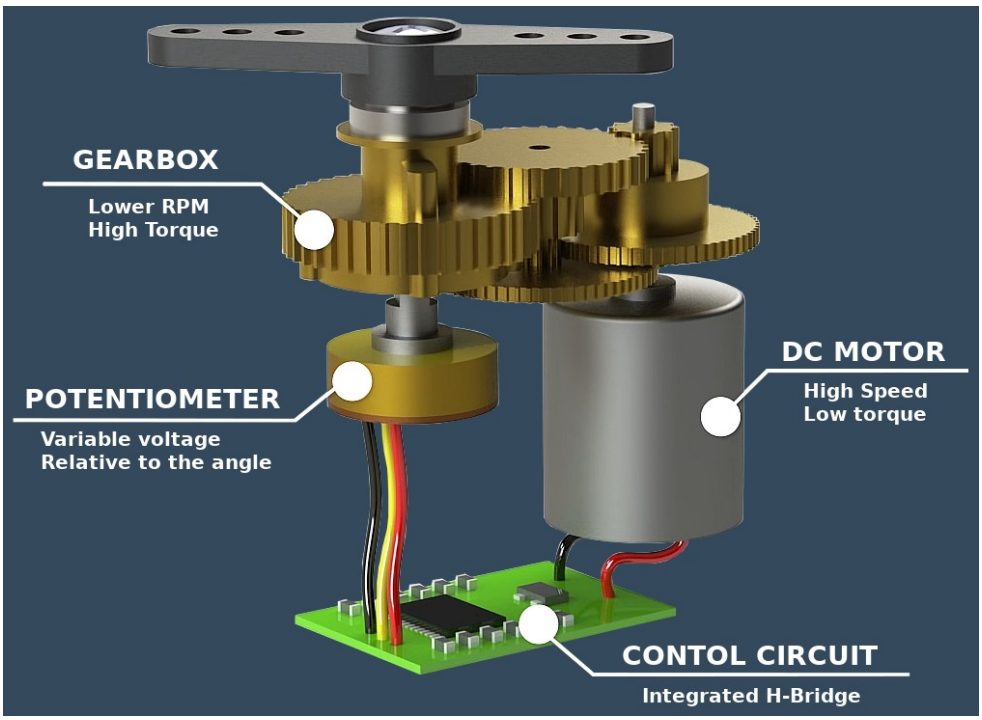
Welcome!

Thank you very much for purchasing our AZ-Delivery Micro Servo Motor SG90. On the following pages, we will introduce you to how to use and setup this handy device.

Have fun!

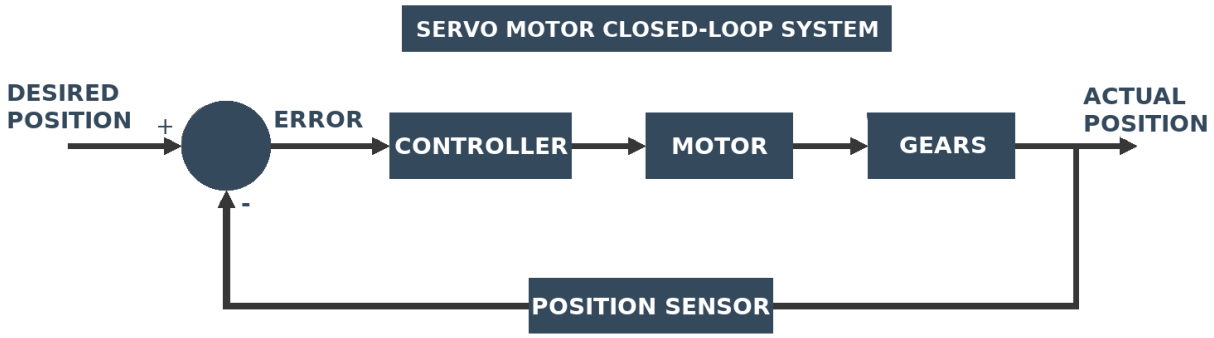


Servo motor introduction



A servo motor is a special device containing a DC motor (high speed, low torque), feedback electronics (integrated H-Bridge) with potentiometer and a set of gears.

The operation of the servo motor can be described by the diagram below:



A servo motor accepts DC voltage in form of pulses (desired position), usually created by microcontroller using PWM (Pulse Width Modulation) and output shaft rotation to the desired position.

When the DC motor is at stop, it means the motor shaft and potentiometer knob are in its equilibrium position. Input DC voltage is same as voltage from potentiometer (via voltage divider) so the inputs to the error amplifier (circle on the image above) are equal and the motor is at stop.

When different voltage is applied, the two inputs on the error amplifier is no longer equal and thus the amplifier drives the motor. As the motor turns, the gear assembly turns and thus also turns the potentiometer, lowering the difference between input voltage and potentiometer voltage. The potentiometer generates a voltage equivalent to that of the input pulse that stops the DC motor, indicating equilibrium.

The gear assembly slows down the rotation of the motor to a speed that the potentiometer can catch up. Also, the gears boost the torque output of the servo motor.

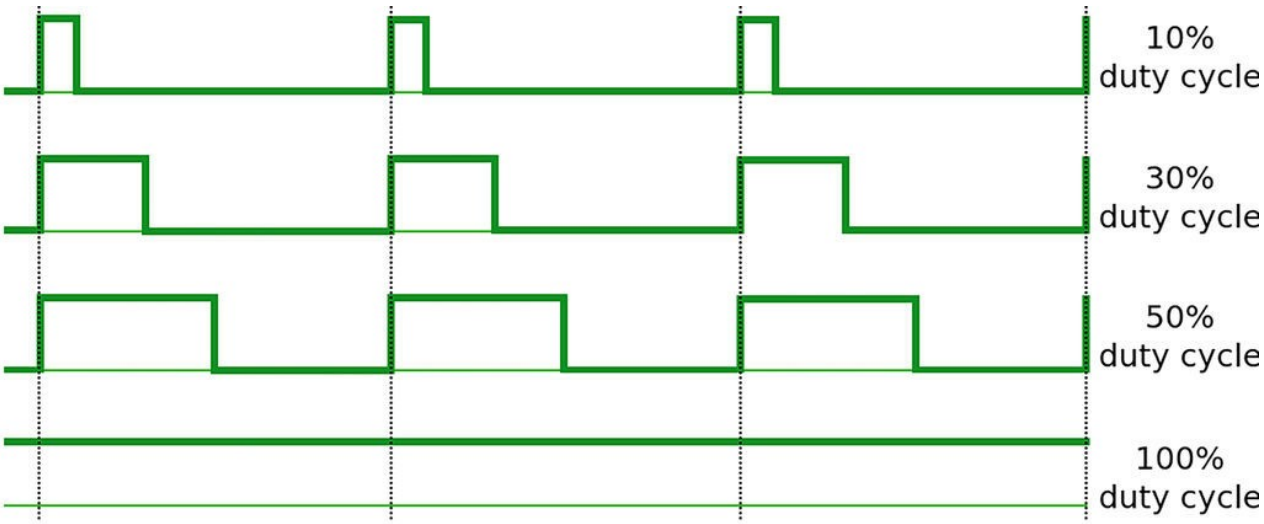
PWM – Pulse Width Modulation

PWM toggles the output of microcontroller ON and OFF very quickly, so quickly that motor which is attached to the output of microcontroller can't react fully. The result is that the motor attached to the output sees a voltage that is proportional to the average percent the time that the microcontroller spends with its output on. This reduces average power (average voltage and average current) delivered by output.

Toggling the output ON and OFF very quickly means that output signal, that the motor sees, oscillates on some frequency. Inertia affect rotation of motor shaft, so that motors react slowly to voltage pulses (PWM signal). But there is a limit to this PWM switching frequency, it also has to be high enough not to affect the motor. For all servo motors we sell PWM frequency is 50Hz.

The average time that PWM output spend on 5V, relative to the one period (time spent on 5V plus time spent on 0V in one pulse), is called duty cycle. For example, a duty cycle of 50%, output is 50% on 5V and 50% on 0V, and if the duty cycle is 25% then output is 25% on 5V and 75% on 0V.

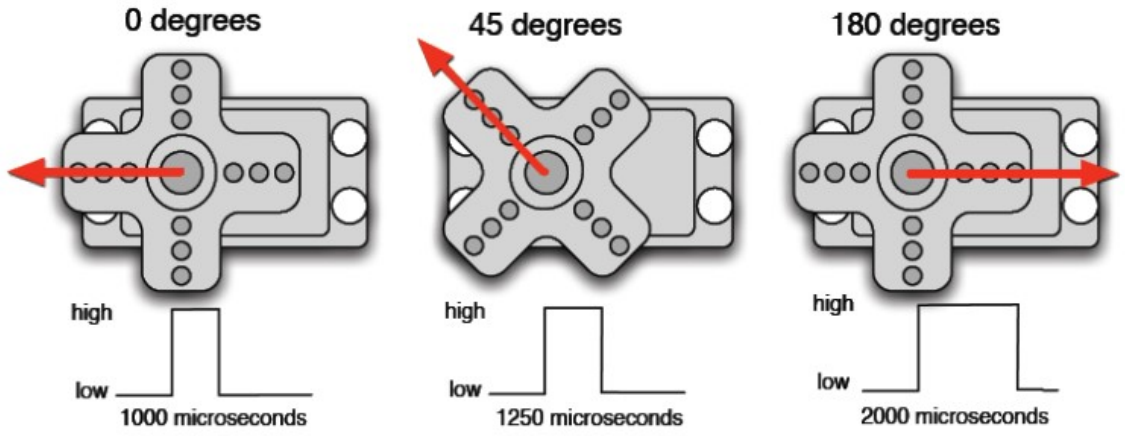
All PWM does is that it simulates analog DC voltage. By changing the duty cycle we change the amplitude of this simulated analog DC voltage. So because the limits of PWM signal are 0V and 5V, duty cycle of 50% is equal to the 2.5V DC, and duty cycle of 25% is equal to the 1.25V DC.



Frequency of 50Hz means that one period of signal lasts for 20ms:
(1 / frequency) = time in seconds

Duty cycle of 25% means, that signal in one period is 5ms on 5V and 15ms on 0V. By changing the duty cycle we change the servo motor arm position.

For example image below (for period of 20ms),
on the left duty cycle is 1ms = 1000us, which is 5%,
in the middle duty cycle is 1.25ms = 1250us, which is 6.25%, and
on the right, duty cycle is 2ms = 2000us, which is 10%.



The exact duty cycle could vary per servo motor. If you want to get the exact pulse width for a specific angle, you first must test your servo motor.

For all servo motors we sell, arm can move from 0° to 180°. That is the limit made in gearbox inside the servo motor. You can change this, there are many posts on the internet explaining how to make servo motor spin 360°, but be careful, you can destroy servo motor in process, and all changes made are irreversible!

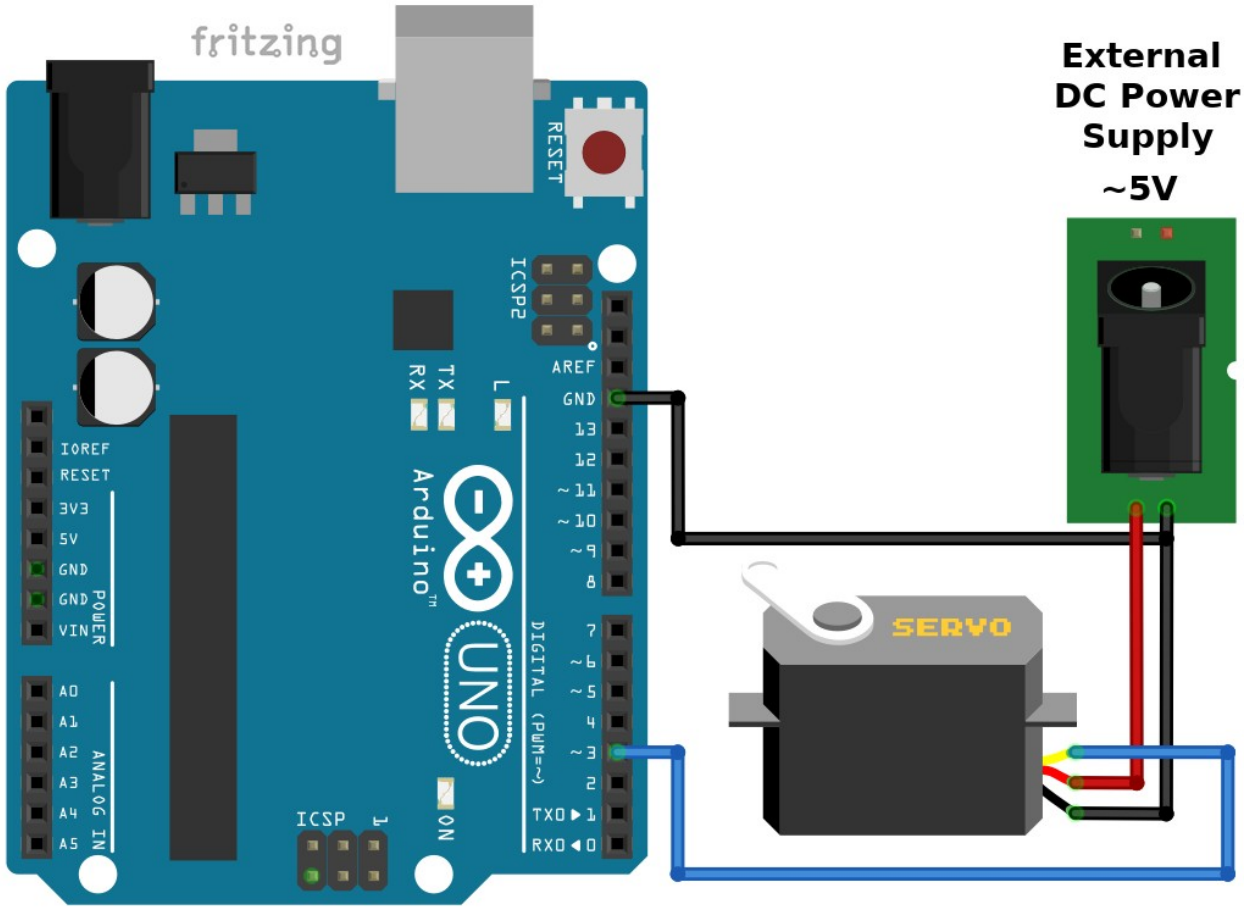
Technical specifications of MG996R

- » Operating voltage: 4.8V - 6V
- » Stall torque: 1.8kgf*cm (4.8V)
- » Operating speed: 0.12s/60º (4.8V)
- » Running current: 100mA - 250mA
- » Stall current: 360mA (6V)
- » Dead band width: 1µs
- » Rotation: 0º - 180º
- » Temperature range: 0ºC – 55ºC
- » Weight: 9g
- » Dimensions: 23x13x26mm
- » Cable length: 24cm

Stall Torque is the torque that is produced by a servo motor when the output rotational speed is zero or the torque load that causes the output rotational speed of a servo motor to become zero - i.e. to cause stalling. 1.8 kgf * cm (kilograms force times centimeters) means that 1.8kg can be hanged on arm end, that is 1cm long, attached to the motor shaft to stop rotation of motor shaft. If arm is 2cm long, half of a load (0.9kg) can be hanged on its end to stop the motor shaft rotation, etc. And stall current in this situation will be 360mA.

It is recommendable to power servo motor from external power supply not from Arduino or Raspberry Pi boards even though the Arduino (or Raspberry Pi) board is capable to drive the servo motor. It is better to have separate power supply for the logic and for the actuators!

Connectiong the servo motor with Arduino



Servo pin	**>**	**Arduino pin**	
Yellow pin	>	D3	**Blue wire**
Brown (or black) pin	>	GND	**Black wire**

Servo pin	**>**	**External Power supply**	
Red pin	>	4.8V - 6V	**Red wire**
Brown (or black) pin	>	GND or 0V	**Black wire**

You can connect yellow pin of the servo motor to any PWM pin on Arduino Uno board. There are six PWM outputs, labeled with "~" sign in the name of digital output: 3, 5, 6, 9, 10 and 11.

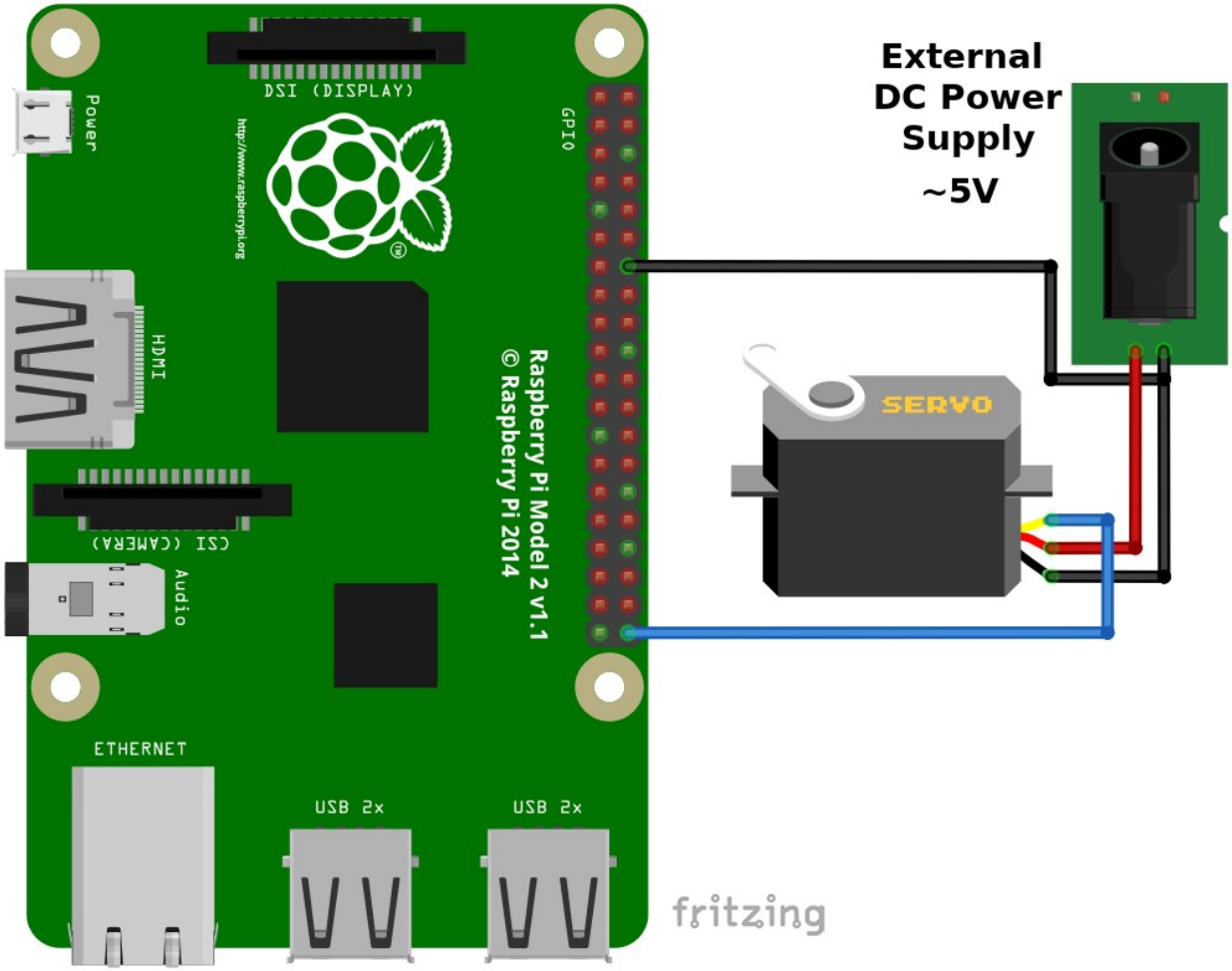
Arduino Sketch

The library for servo motors, comes preinstalled with Arduino IDE. It is called "*Servo.h*" and with it comes two sketch examples. We will use "*Sweep*" example. To open it, go to *File* > *Examples* > *Servo* > *Sweep*. Sketch is self explanatory, so we won't go into details. Here is the sketch:

```
#include <Servo.h>
Servo myservo;  // create servo object to control a servo
// twelve servo objects can be created on most boards
int pos = 0;    // variable to store the servo position
void setup() {
  myservo.attach(3);  // attaches the servo on pin 3 to the servo object
}

void loop() {
  for (pos = 0; pos <= 180; pos += 1) {
    // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);
    delay(15);
  }
  for (pos = 180; pos >= 0; pos -= 1) {
    // goes from 180 degrees to 0 degrees
    myservo.write(pos);
    delay(15);
  }
}
```

Connectiong the servo motor with Raspberry Pi



Servo motor pin	**>**	**Raspberry Pi pin**	
Yellow pin	>	GPIO21 [pin 40]	**Blue wire**
Brown (or black) pin	>	GND [pin 14]	**Black wire**

Servo motor pin	**>**	**External Power supply**	
Red pin	>	4.8V - 6V	**Red wire**
Brown (or black) pin	>	GND or 0V	**Black wire**

Script example

If you don't already have installed "RPi.GPIO" library, here is how to do it. Start your Raspberry Pi, and open terminal, then run these commands. First you have to update Raspbian by running:

```
sudo apt-get update && sudo apt-get upgrade -y
```

Then you are ready to install "RPi.GPIO" library. Run this command to install it:

```
sudo apt-get install rpi.gpio
```

After this we are ready to write script. Here is the code:

```
import RPi.GPIO as GPIO
import time

servo = 21 # we connected servo yellow pin to the GPIO21

time_pause = 0.05

GPIO.setmode(GPIO.BCM)
GPIO.setup(servo, GPIO.OUT)

p = GPIO.PWM(servo, 50) # 50hz frequency
p.start(2.0) # starting duty cycle
# (it set the servo to 0 degree)
```

```
def changeDT(x):
    if x == 2.0:
        p.ChangeDutyCycle(x)
        print("{} = 0 degrees".format(x))
    elif x == 7.0:
        p.ChangeDutyCycle(x)
        print("{} = 90 degrees".format(x))
    elif x == 11.5:
        p.ChangeDutyCycle(x)
        print("{} = 180 degrees".format(x))
    else:
        p.ChangeDutyCycle(x)
        print("{}".format(x))

print("[press ctrl+c to end the script]")
try:
    while True:
        x = 2.0
        for k in range(19):
            changeDT(x)
            x += 0.5
            time.sleep(time_pause)
        x = 11.5
        for k in range(19):
            changeDT(x)
            x -= 0.5
            time.sleep(time_pause)

except KeyboardInterrupt:
    p.stop()
    GPIO.cleanup()
```

In order to calculate duty cycle we need to do the following:
Because the frequency is 50Hz, one period lasts for 20ms.
Duty cycle of 0.4ms is equal to the :
(0.4ms / 20ms) * 100 = 2% - and this is equal to 0° position of arm connected to the servo motor.

Duty cycle of 1.4ms is equal to the:
(1.4ms / 20ms) * 100 = 7% - and this is equal to 90° position of arm connected to the servo motor.

Duty cycle of 2.3ms is equal to the:
(2.3ms / 20ms) * 100 = 11.5% - and this is equal to 180° position of arm connected to the servo motor.

Don’t use *"time_pause"* less than 0.05, because if you use 0.04 it is too fast for servo motor to react, so there will be bugs. Servo motor won’t be able to follow position instructions.

Everything else in this script is self explanatory.

You've done it, you can now use your module for your projects.

Now it is time to learn and make the Projects on your own. You can do that with the help of many example scripts and other tutorials, which you can find on the internet.

If you are looking for the high quality products for Arduino and Raspberry Pi, AZ-Delivery Vertriebs GmbH is the right company to get them from. You will be provided with numerous application examples, full installation guides, eBooks, libraries and assistance from our technical experts.

https://az-delivery.de

Have Fun!

Impressum

https://az-delivery.de/pages/about-us