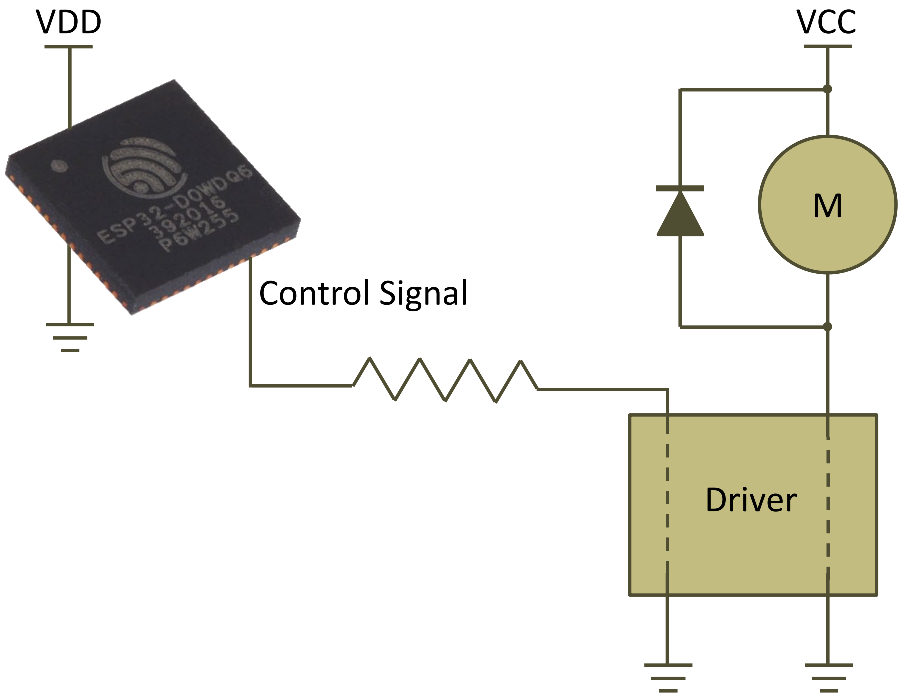
Robust Mechatronics

Actuator Control

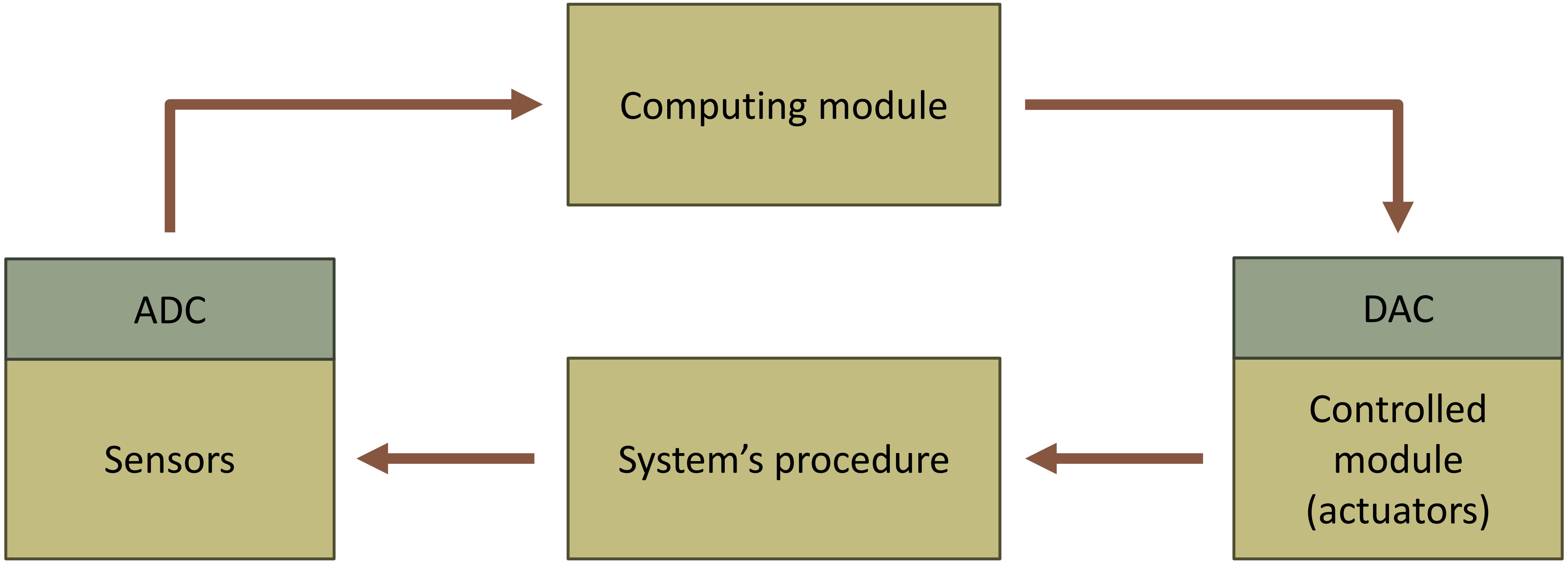


Dr Loukas Bampis, Assistant Professor

Mechatronics & Systems Automation Lab

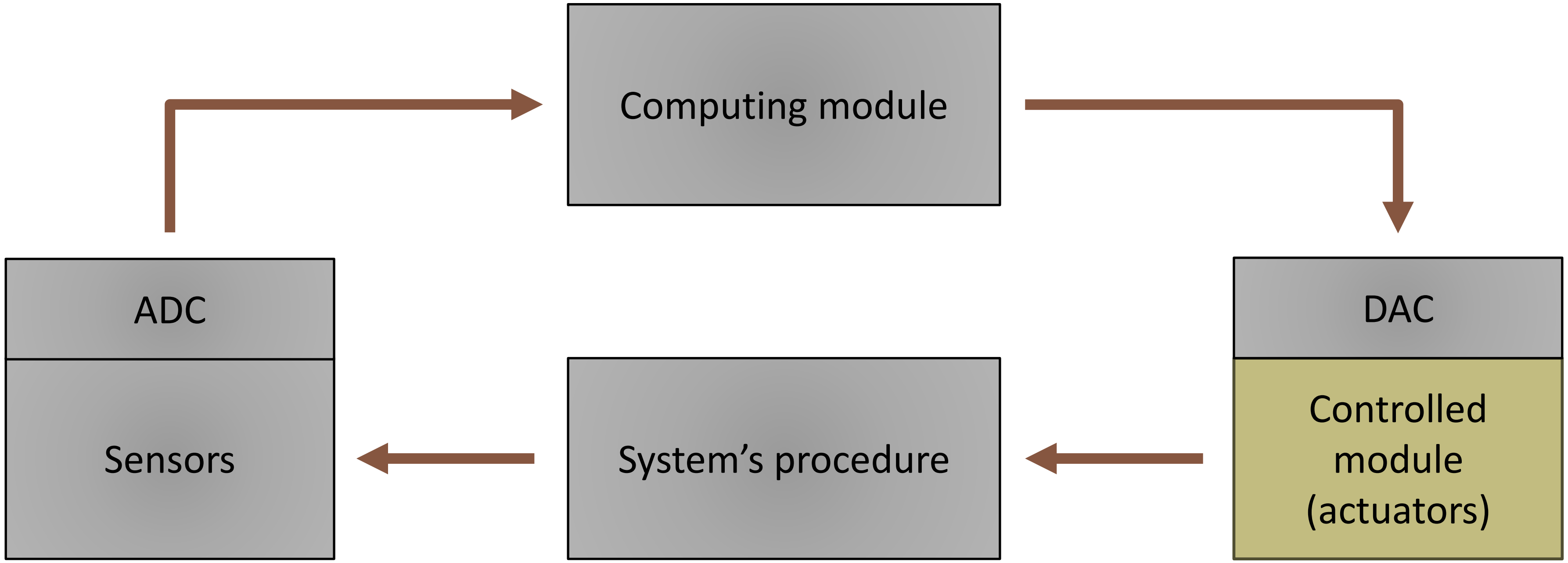
Actuator Control

Control loop



Actuator Control

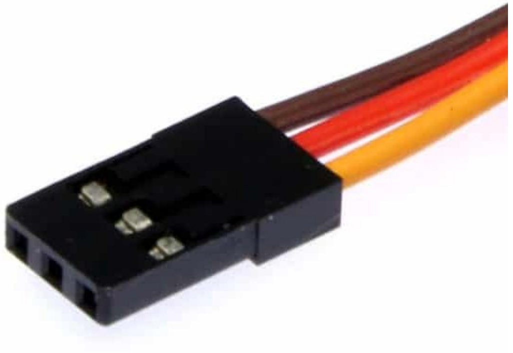
Control loop



Actuator Control

Servo Motors

- They are used in applications where precise control of motor rotation is required (angle or speed)
- Power supply by two terminals
- Rotation adjustment via a separate terminal



Actuator Control

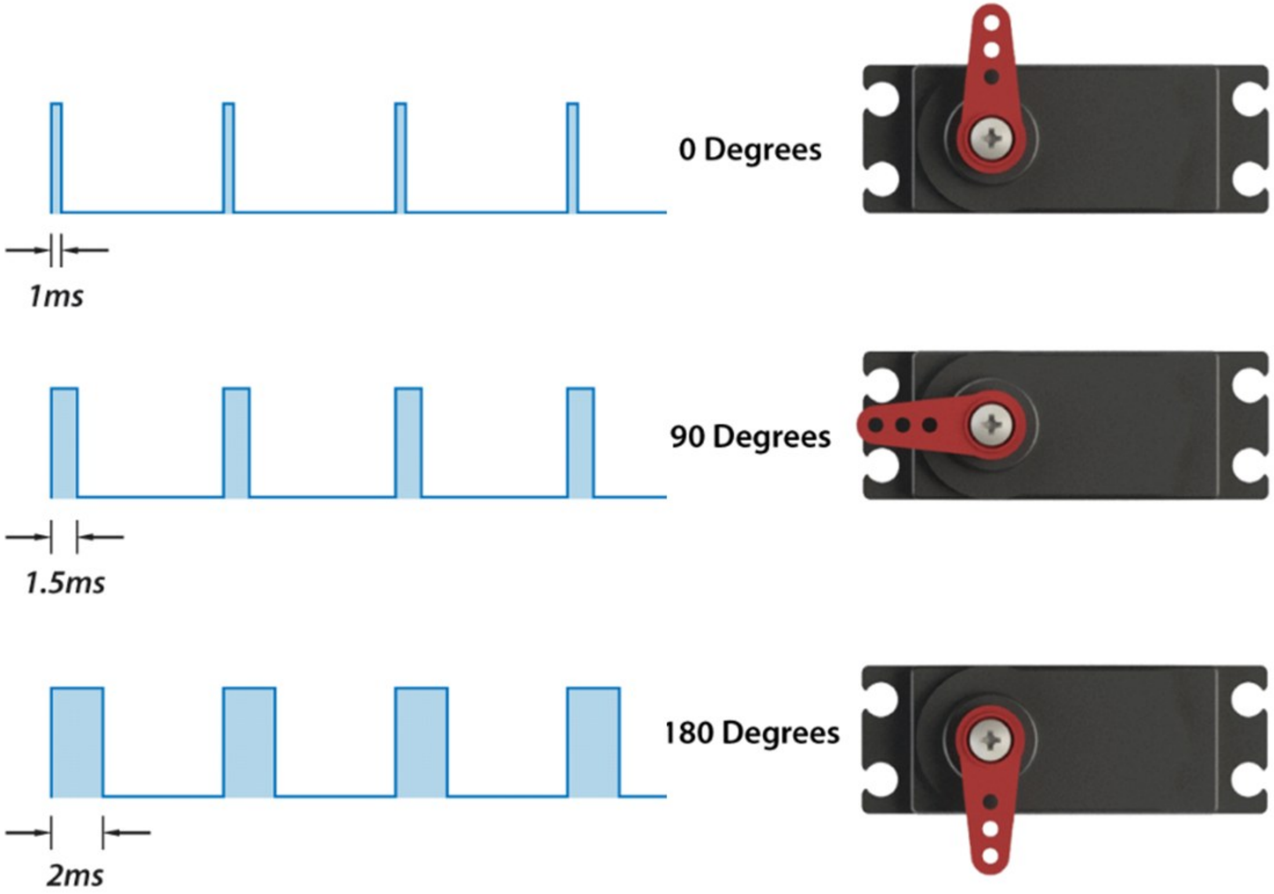
- Control via PWM

- Rotation is not equivalent to the average voltage (like typical PWM), but to the duration of pulses

- In most cases:
  Neutral status: Pulse duration 1.5ms
  - Typical pulse duration range: 1ms έως 2ms
    - 1ms → 0^o
    - 2ms → 180^o
    - 1.5ms → 90^o
  - Linear relationship for pulse duration and angle

- High tolerance to different operating frequencies
  - 1.5ms pulse every 6ms (D=25%)
  - 1.5ms pulse every 25ms (D=6%)
  - Usually a period of 20ms is expected (50Hz)

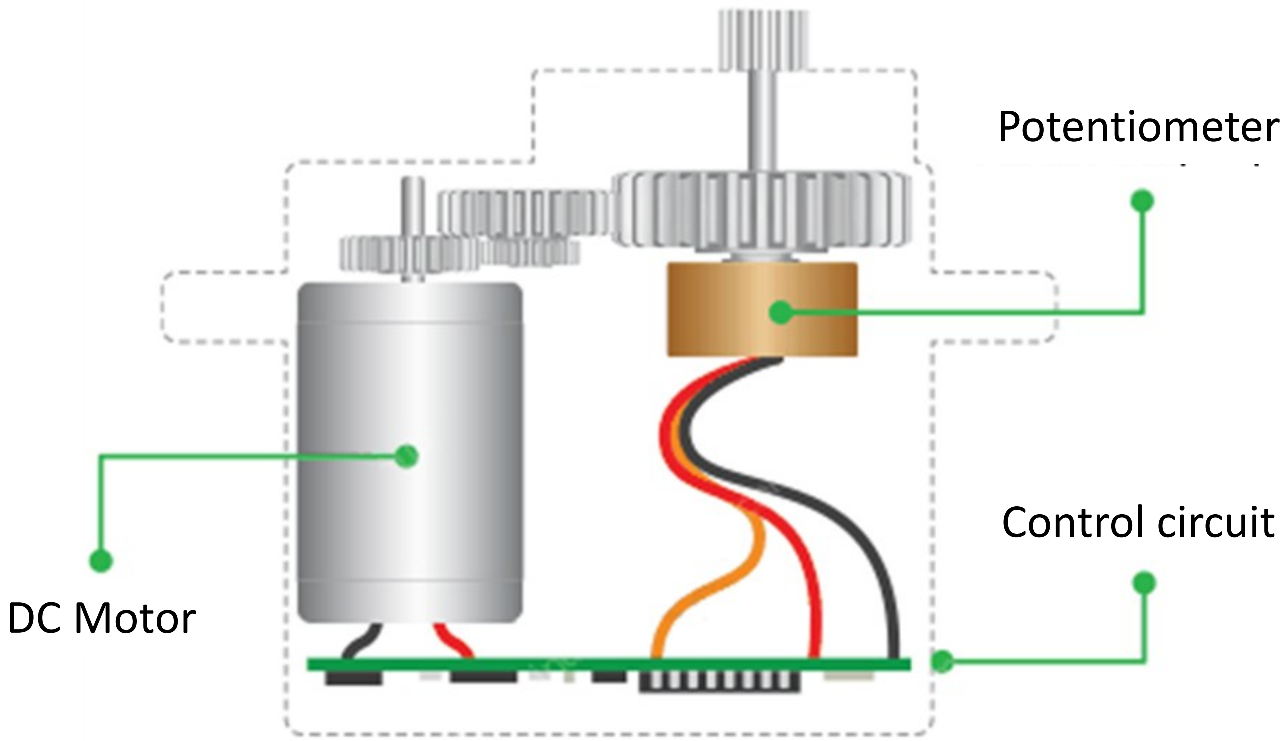
Same rotation effect



Actuator Control

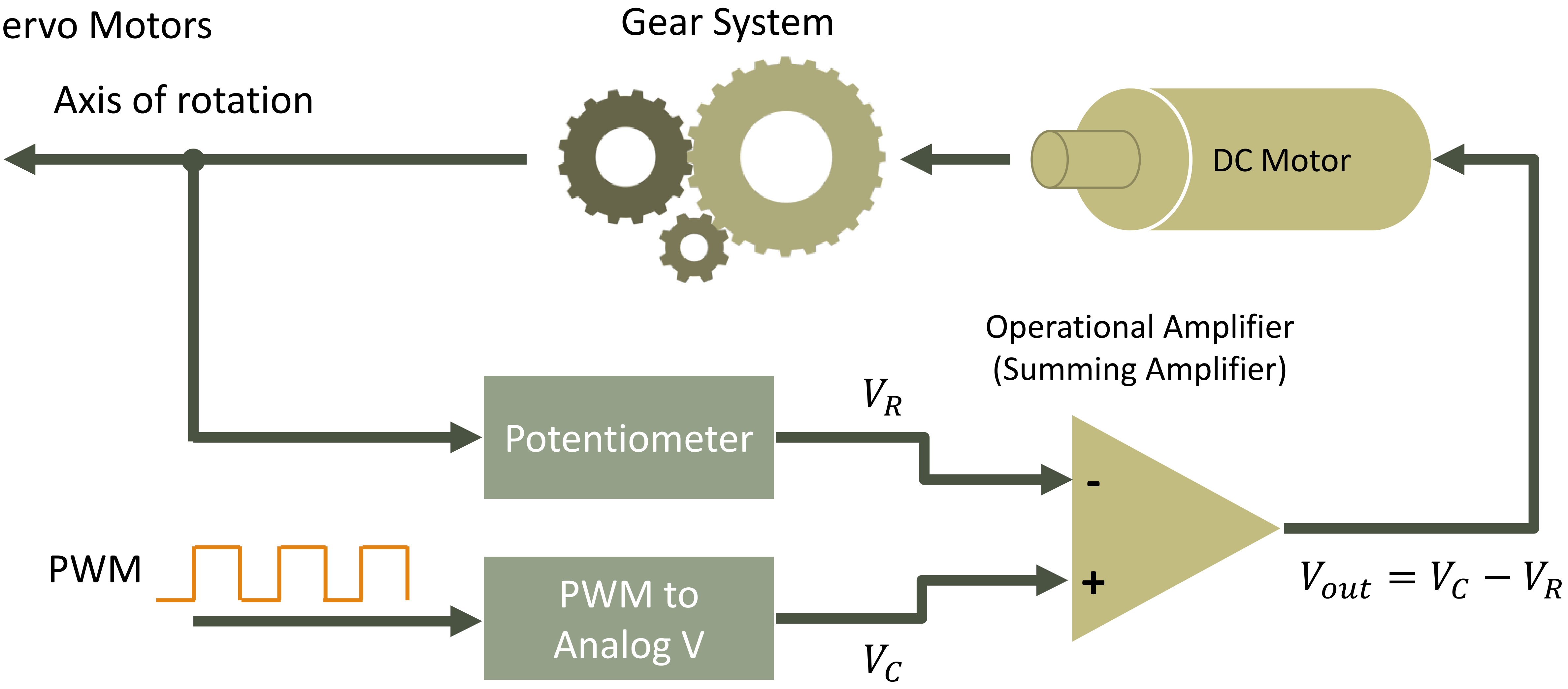
Servo Motors

- Potentiometer connected to the control circuit
- Feedback to identify the angle of rotation of the output
- When the rotation angle reaches the desired value, the voltage supply to the motor is cut off via a Voltage Comparator (Operational Amplifier) and the rotation stops



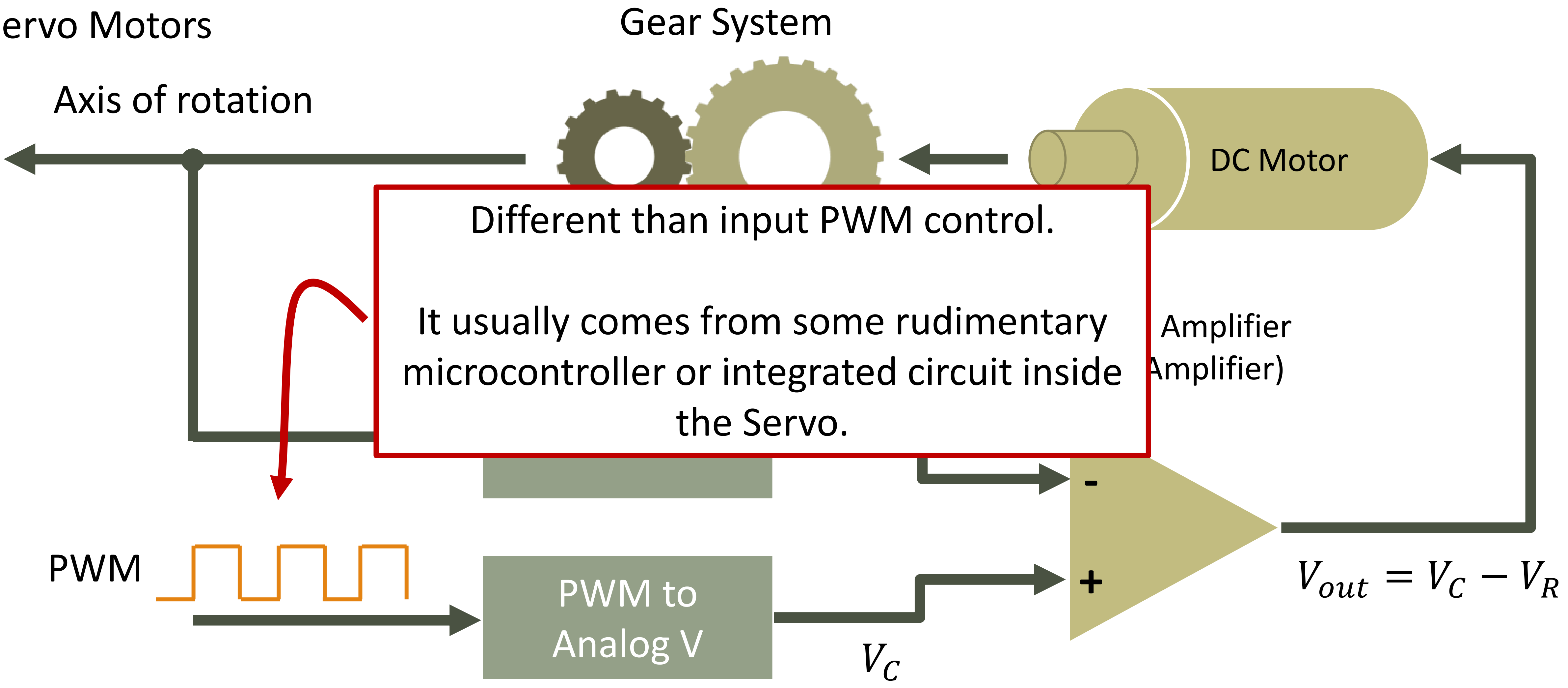
Actuator Control

Servo Motors



Actuator Control

Servo Motors

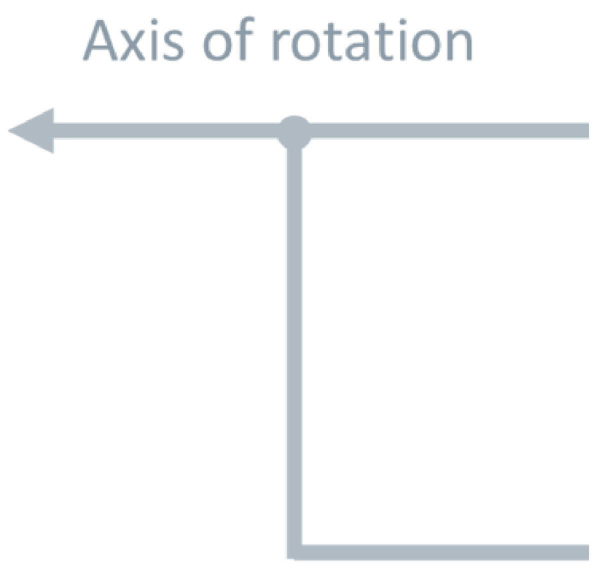


Different than input PWM control.

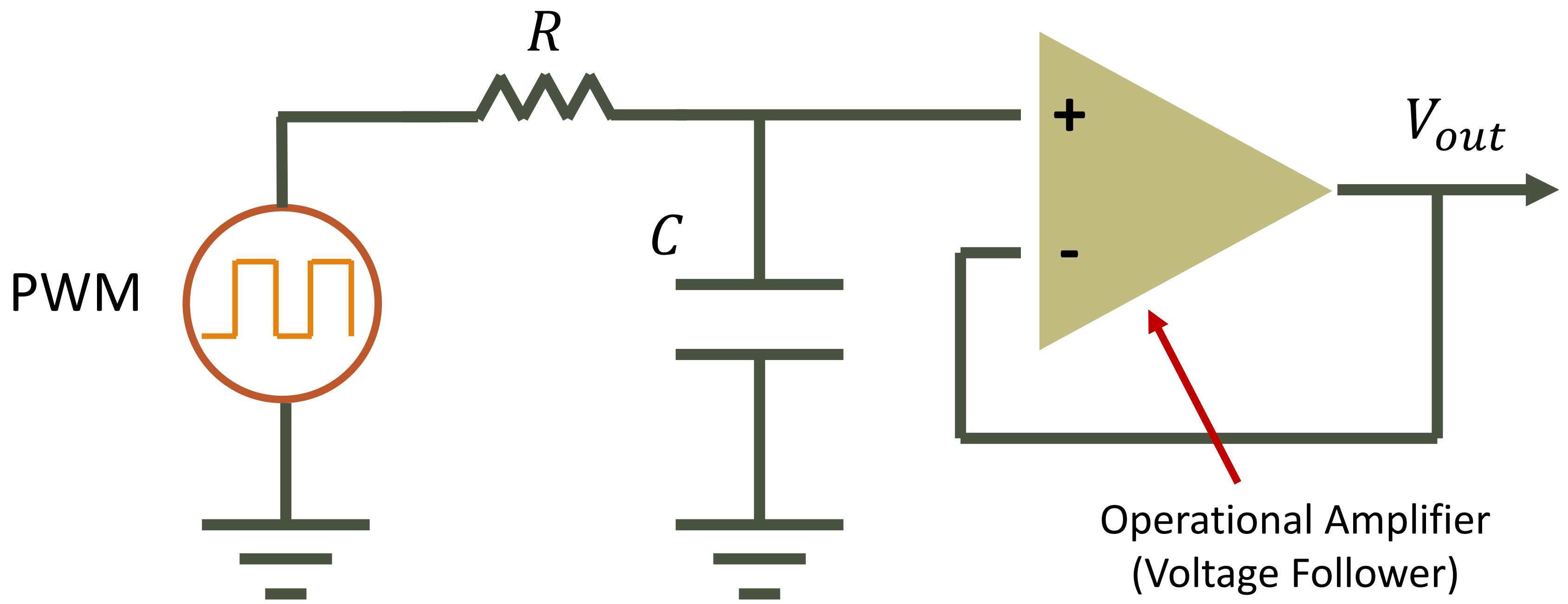
It usually comes from some rudimentary microcontroller or integrated circuit inside the Servo.

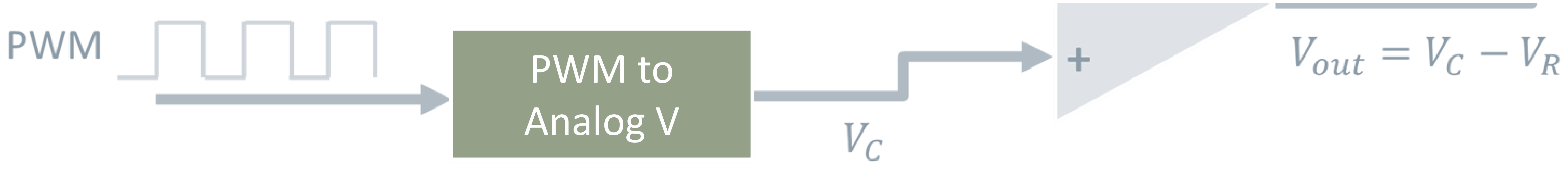
Actuator Control

Servo Motors



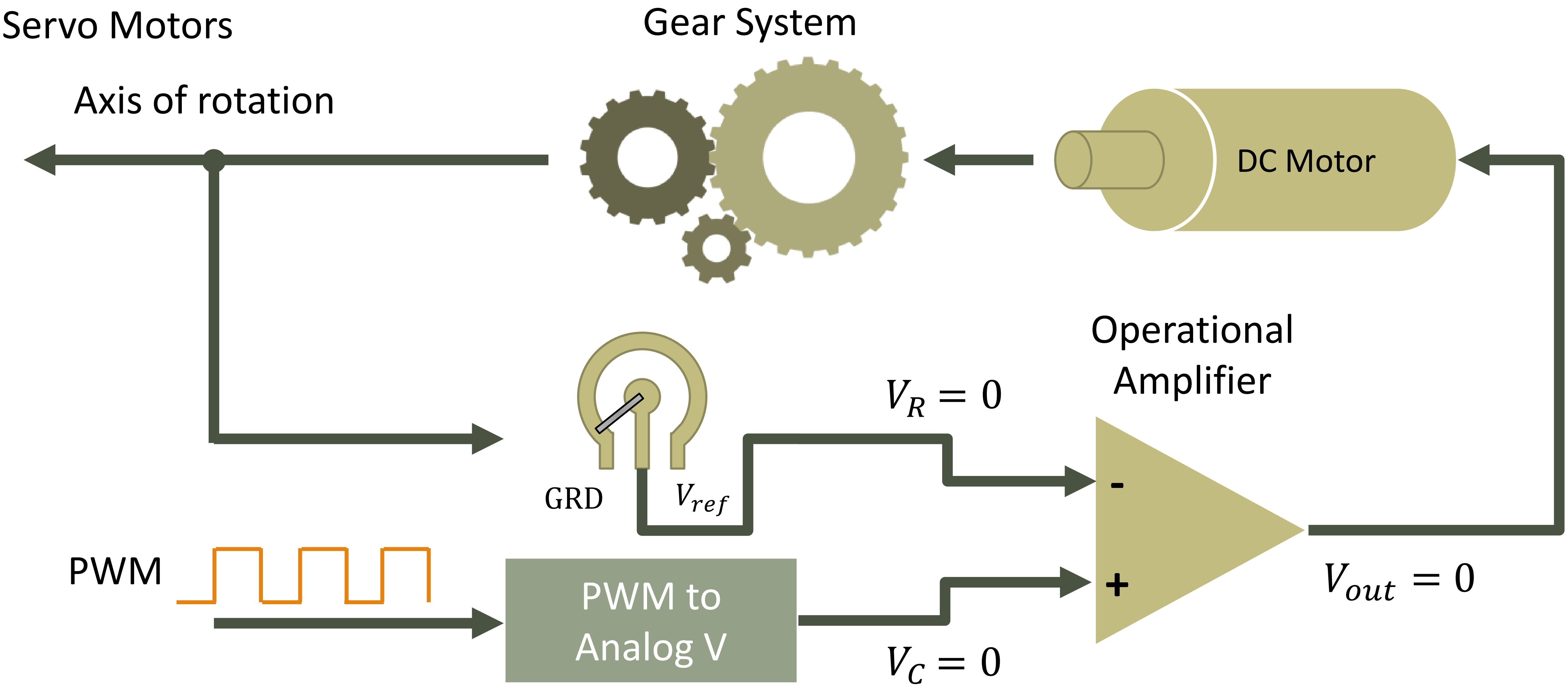
RC Circuit





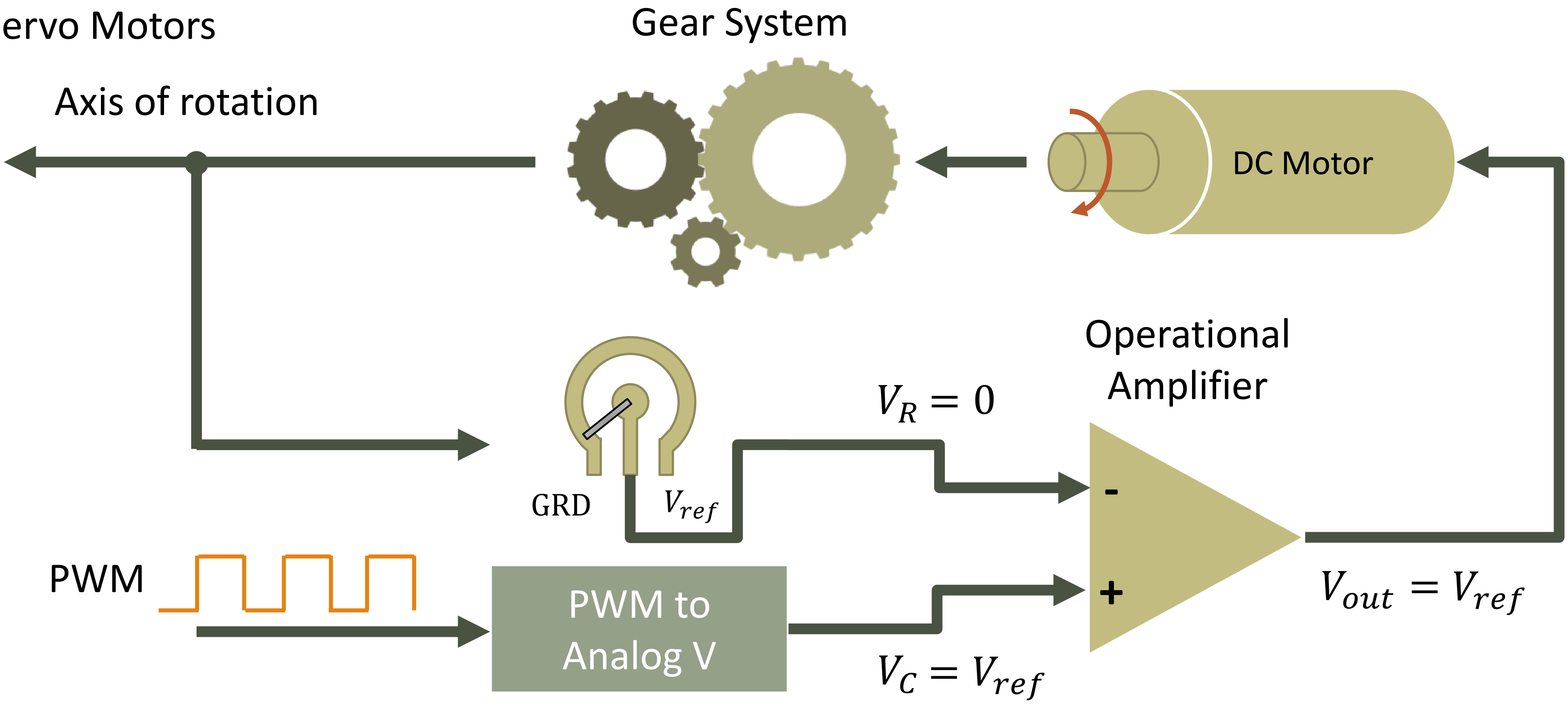
Actuator Control

Servo Motors

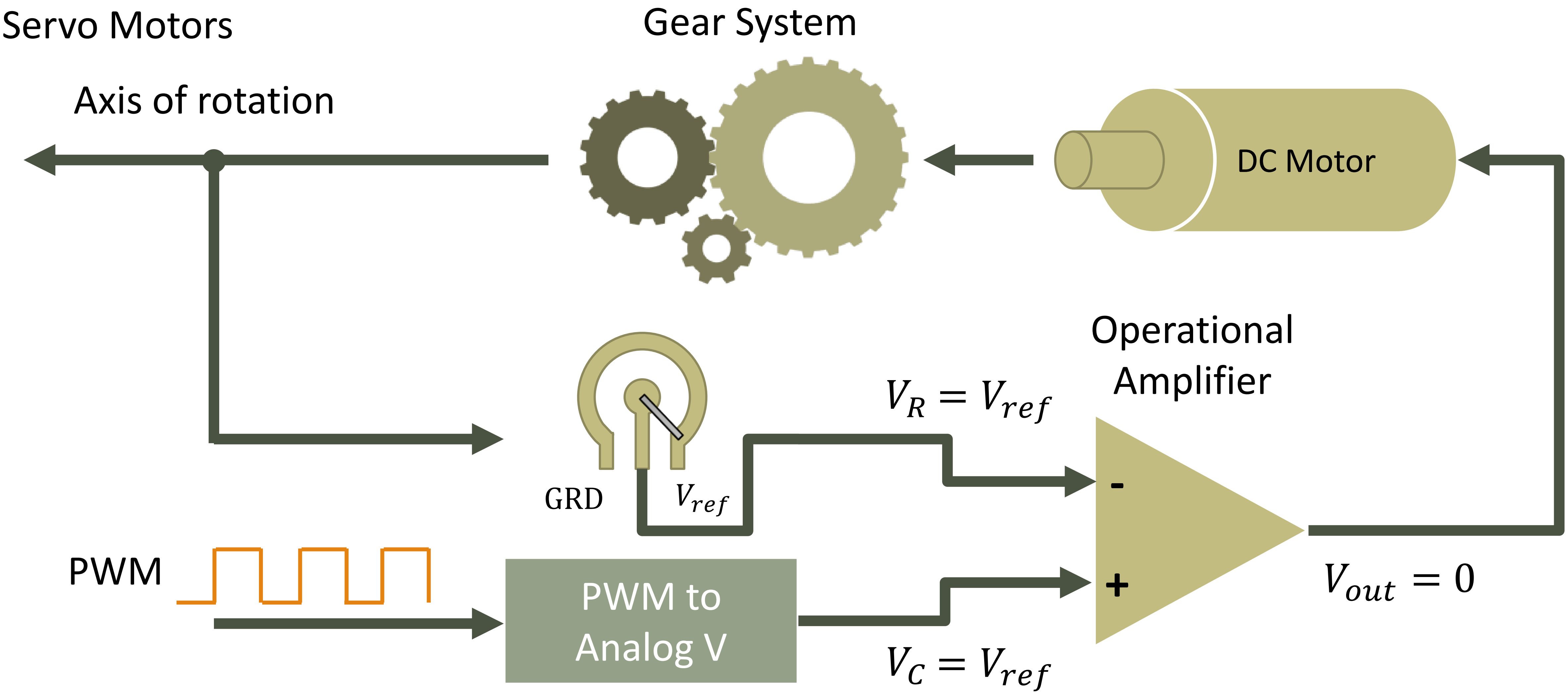


Actuator Control

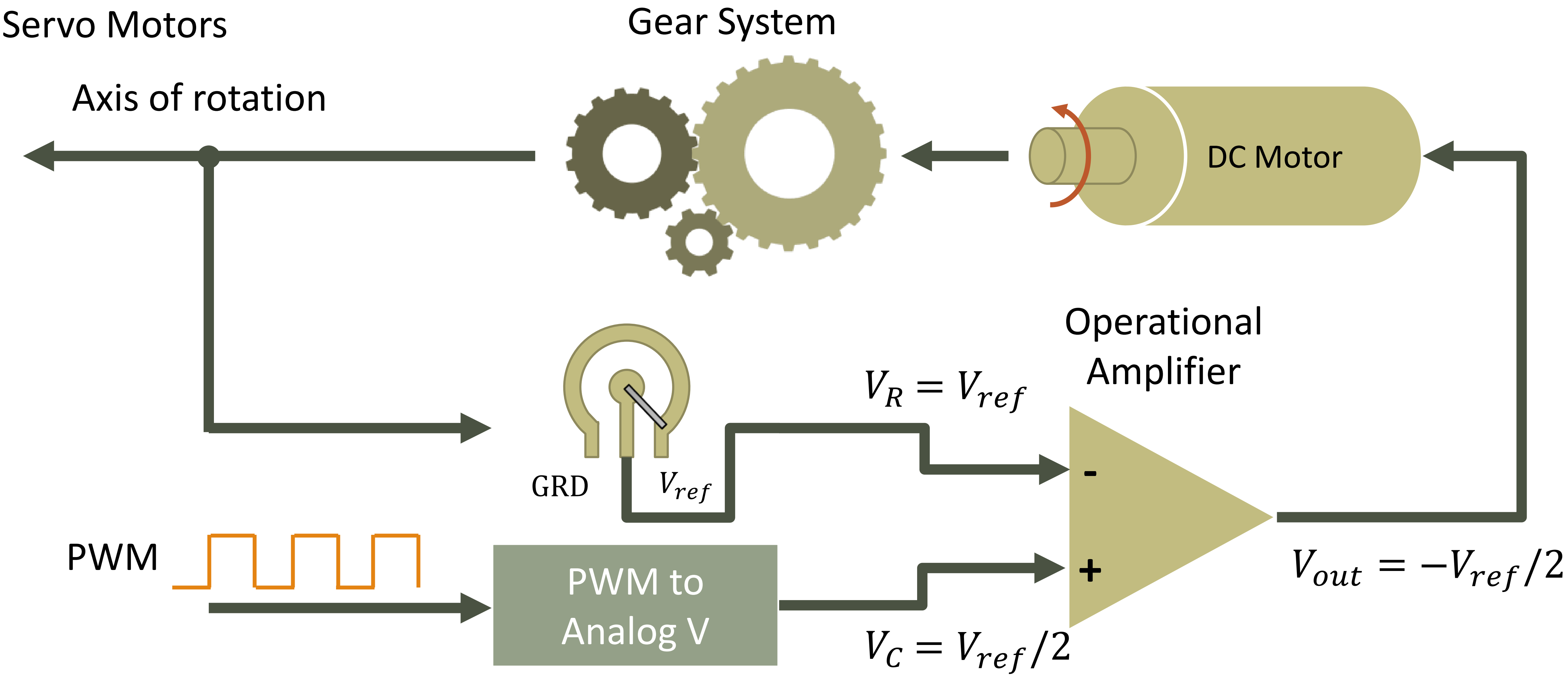
Servo Motors



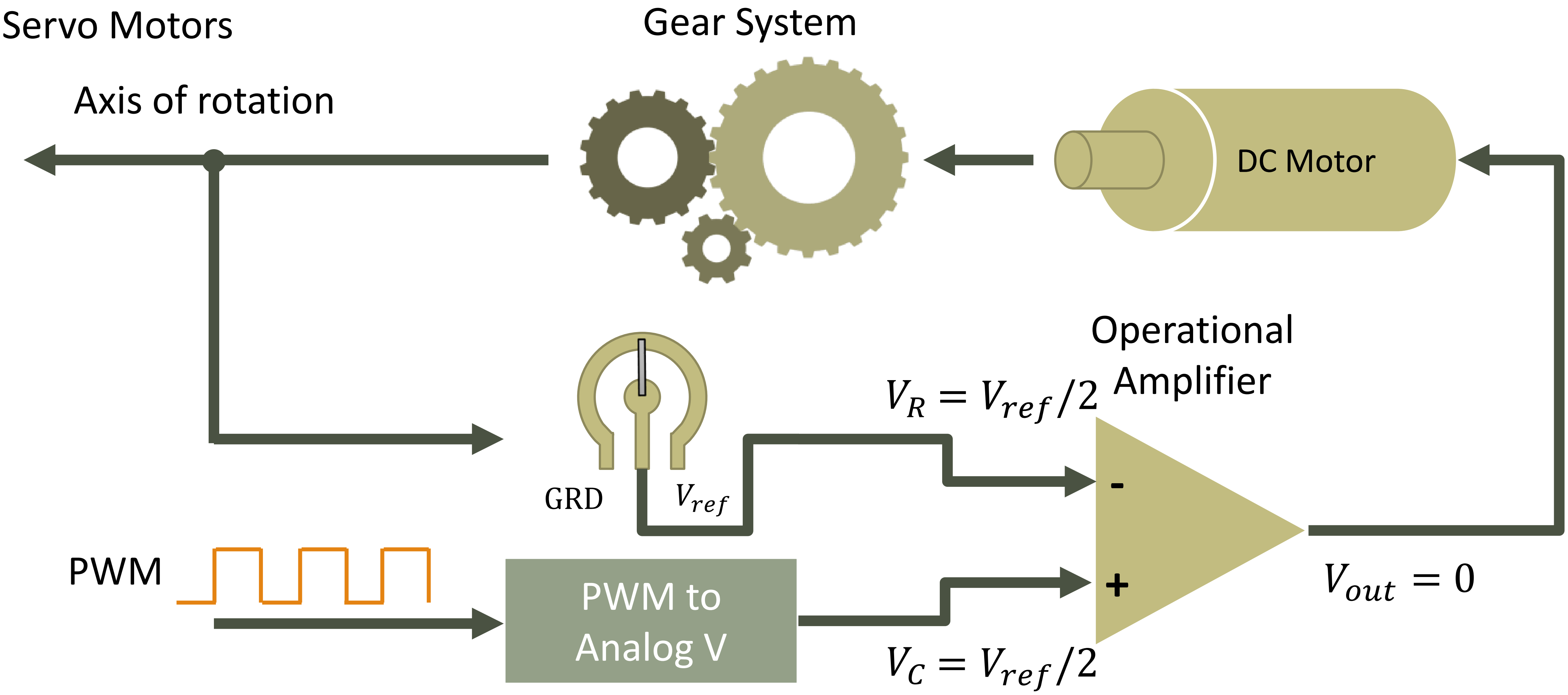
Actuator Control



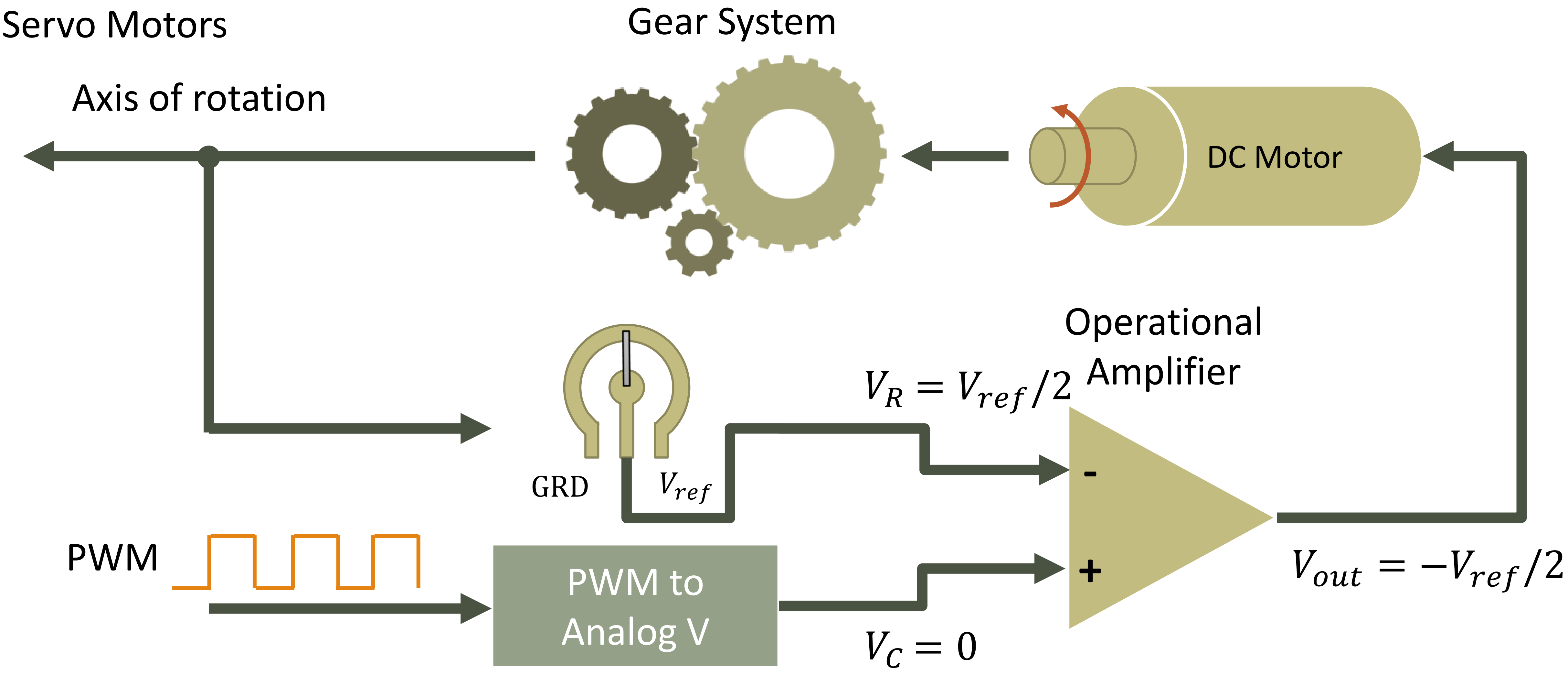
Actuator Control



Actuator Control

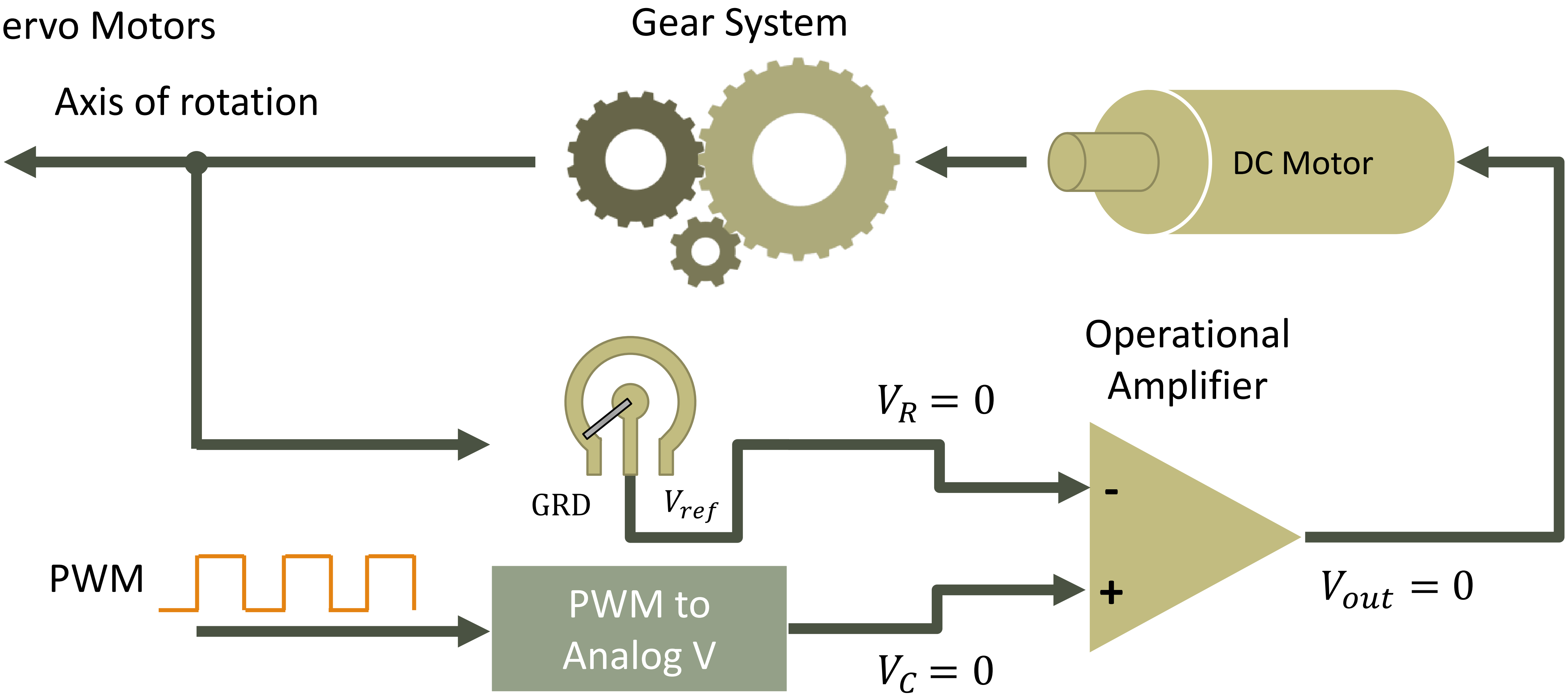


Actuator Control



Actuator Control

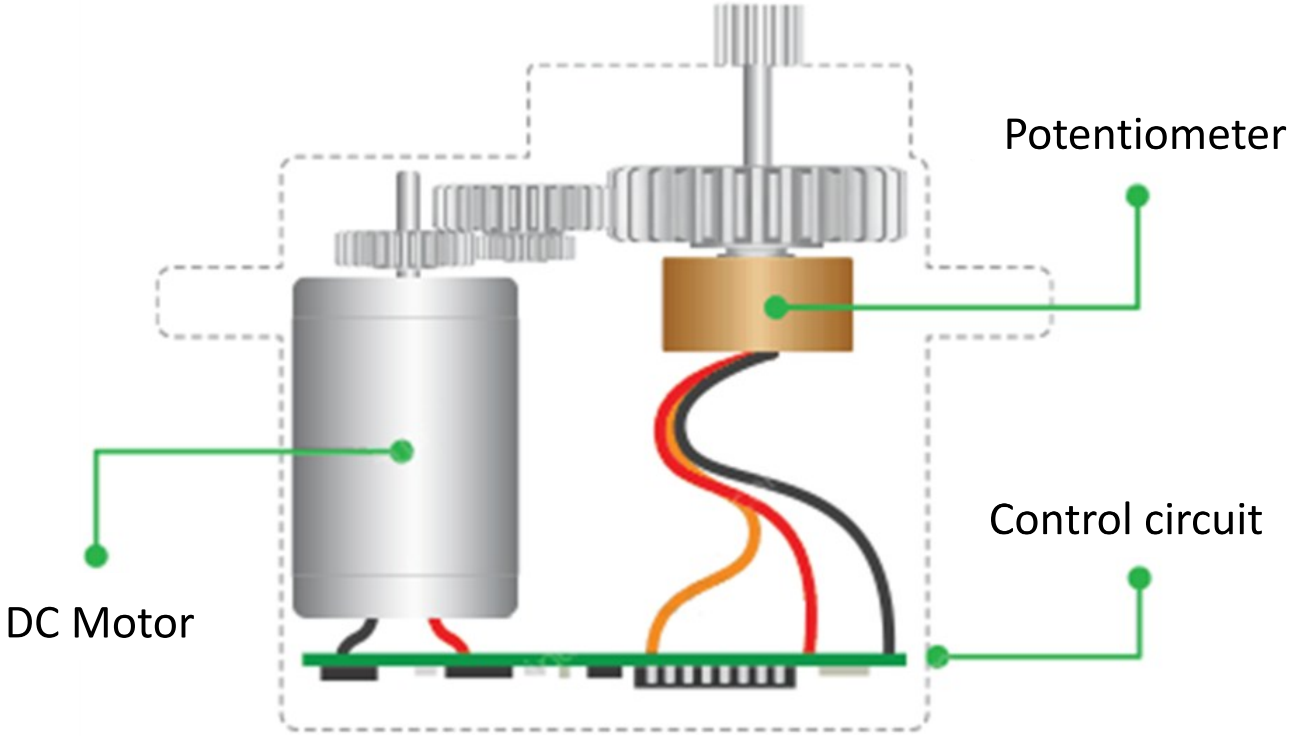
Servo Motors



Actuator Control

Servo Motors

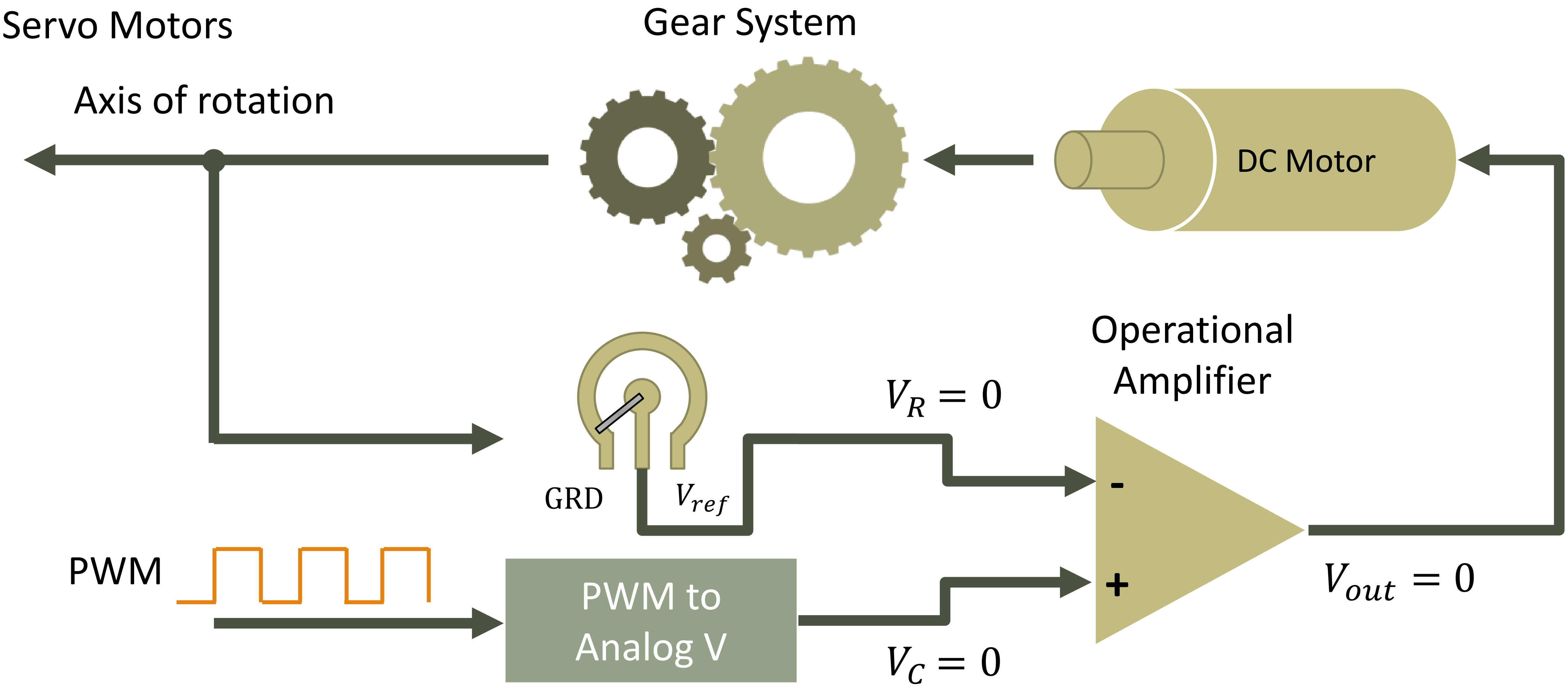
- Potentiometer connected to the control circuit
- Feedback to identify the angle of rotation of the output
- When the rotation angle reaches the desired value, the voltage supply to the motor is cut off via a Voltage Comparator (Operational Amplifier) and the rotation stops



- **Continuous rotation servo (or 360^o servo)**
  - The potentiometer is replaced by a simple voltage divider and disengaged from the output rotation axis
  - The control signal refers to the rotation speed

Actuator Control

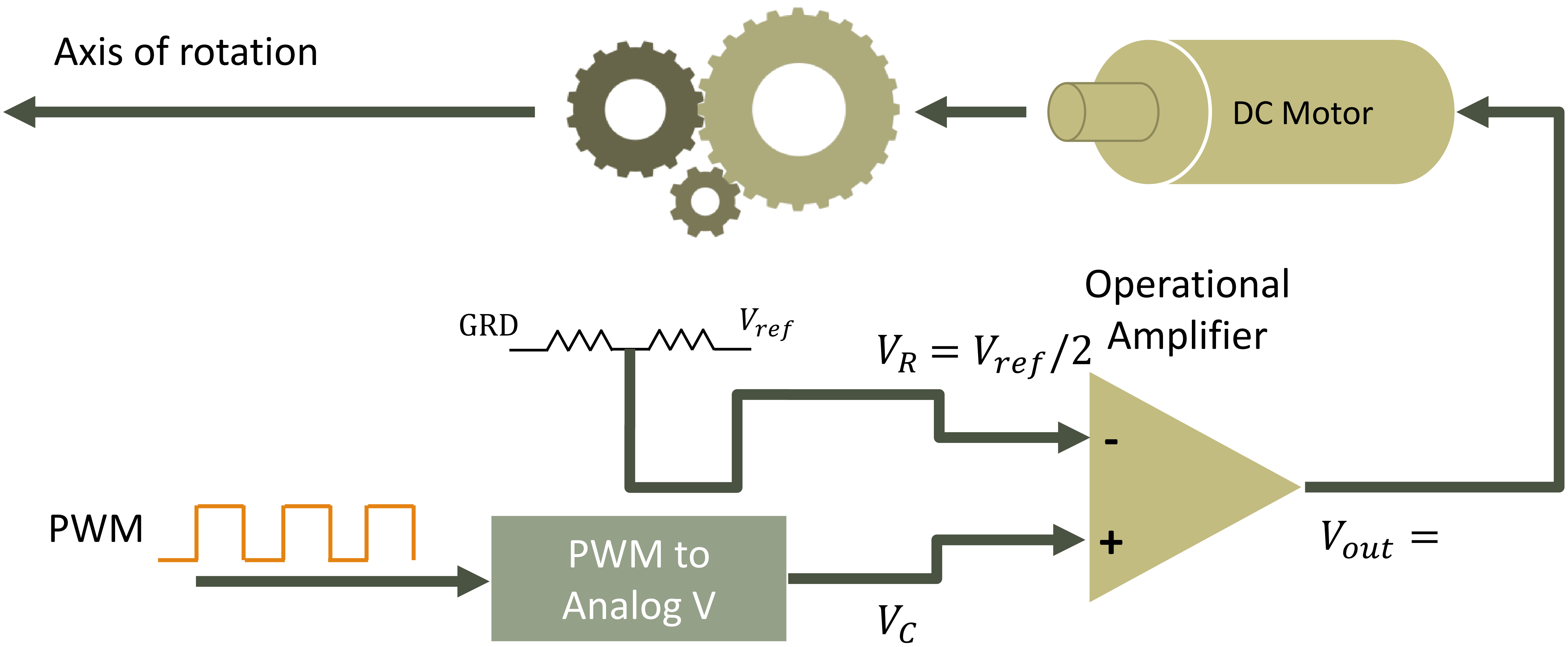
Servo Motors



Actuator Control

Servo Motors

Gear System



Axis of rotation

DC Motor

Operational Amplifier

GRD

V_{ref}

$V_R = V_{ref}/2$

-

+

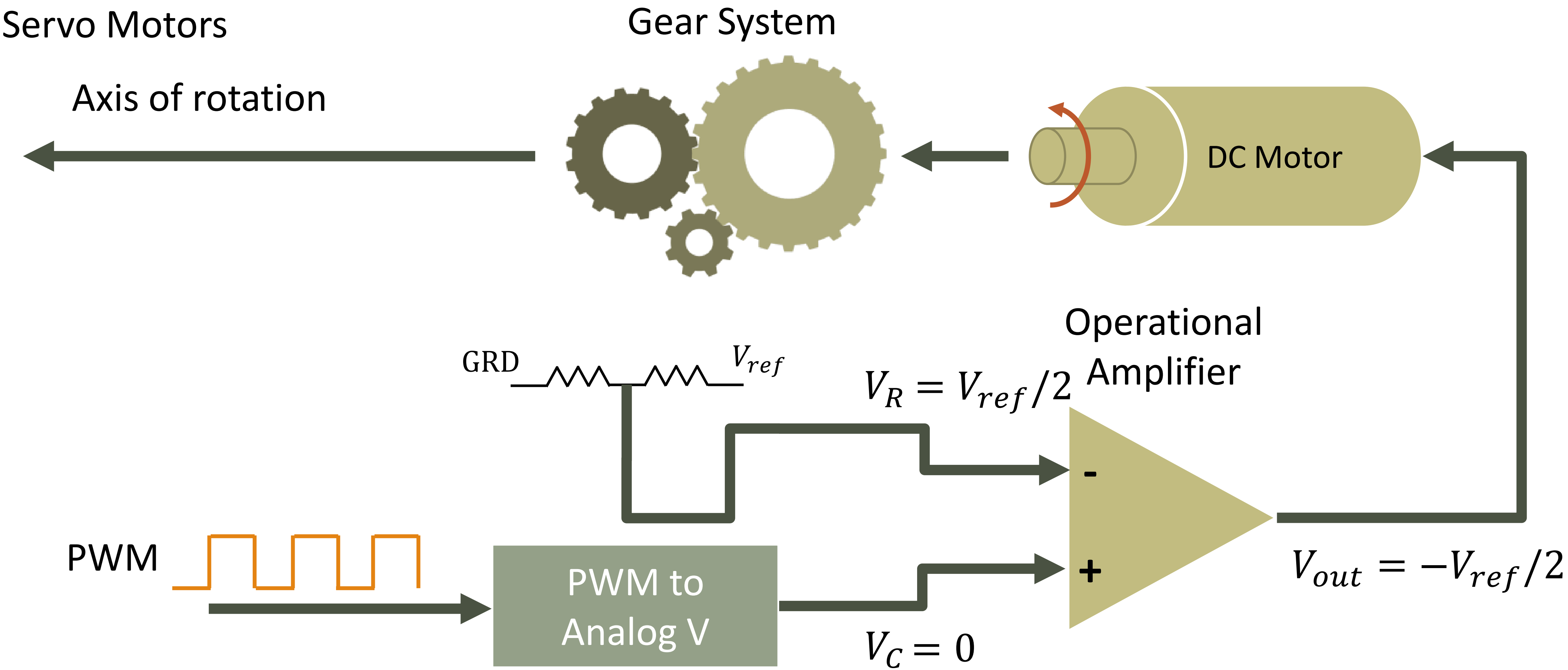
PWM

PWM to Analog V

V_C

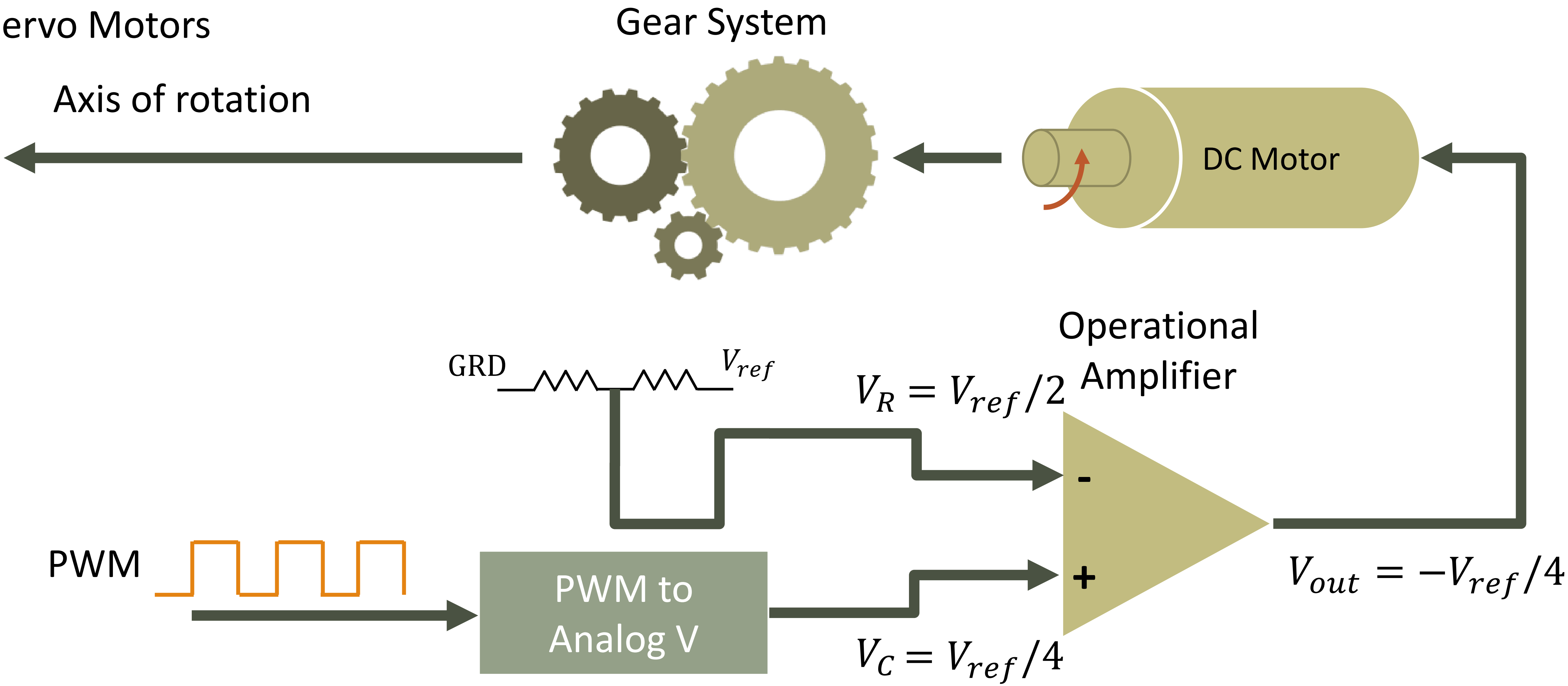
$V_{out} =$

Actuator Control



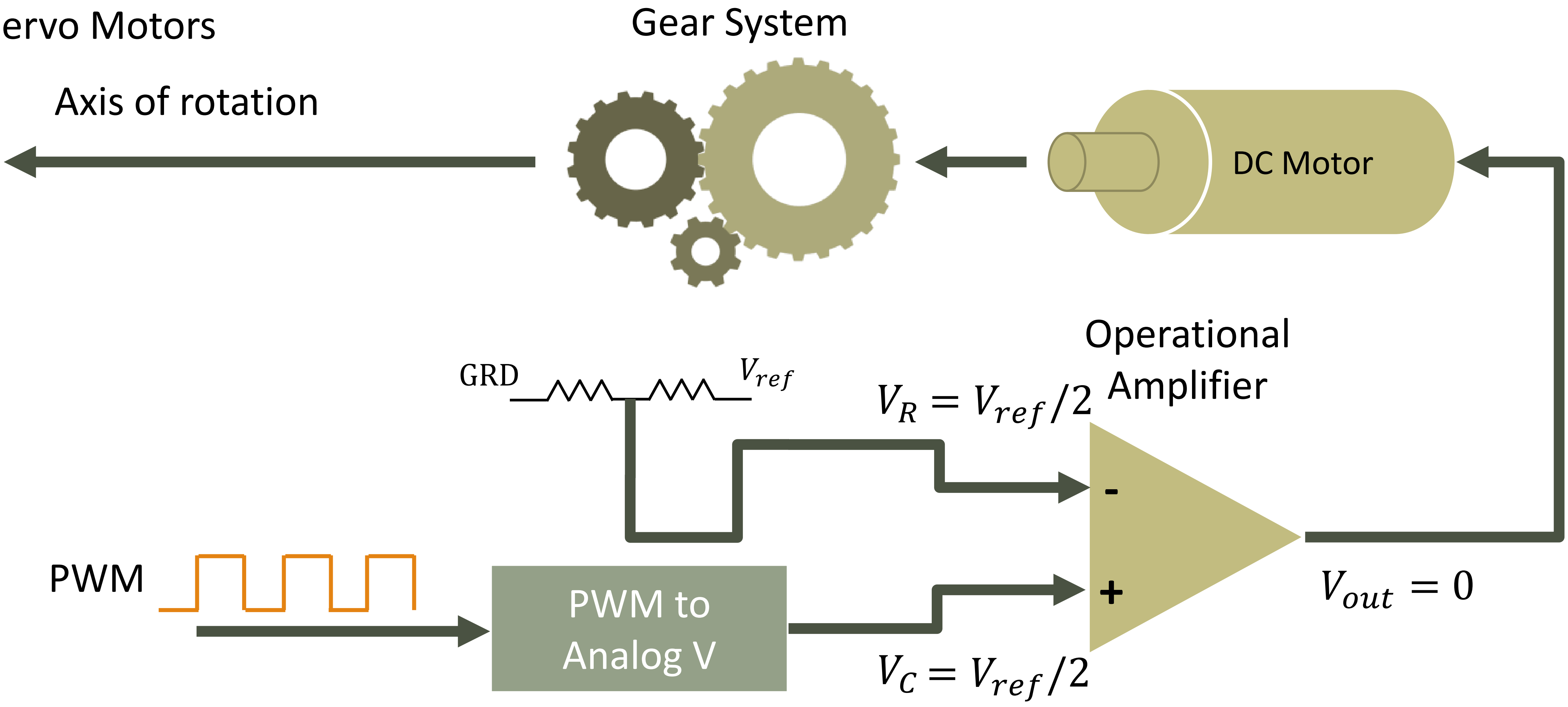
Actuator Control

Servo Motors

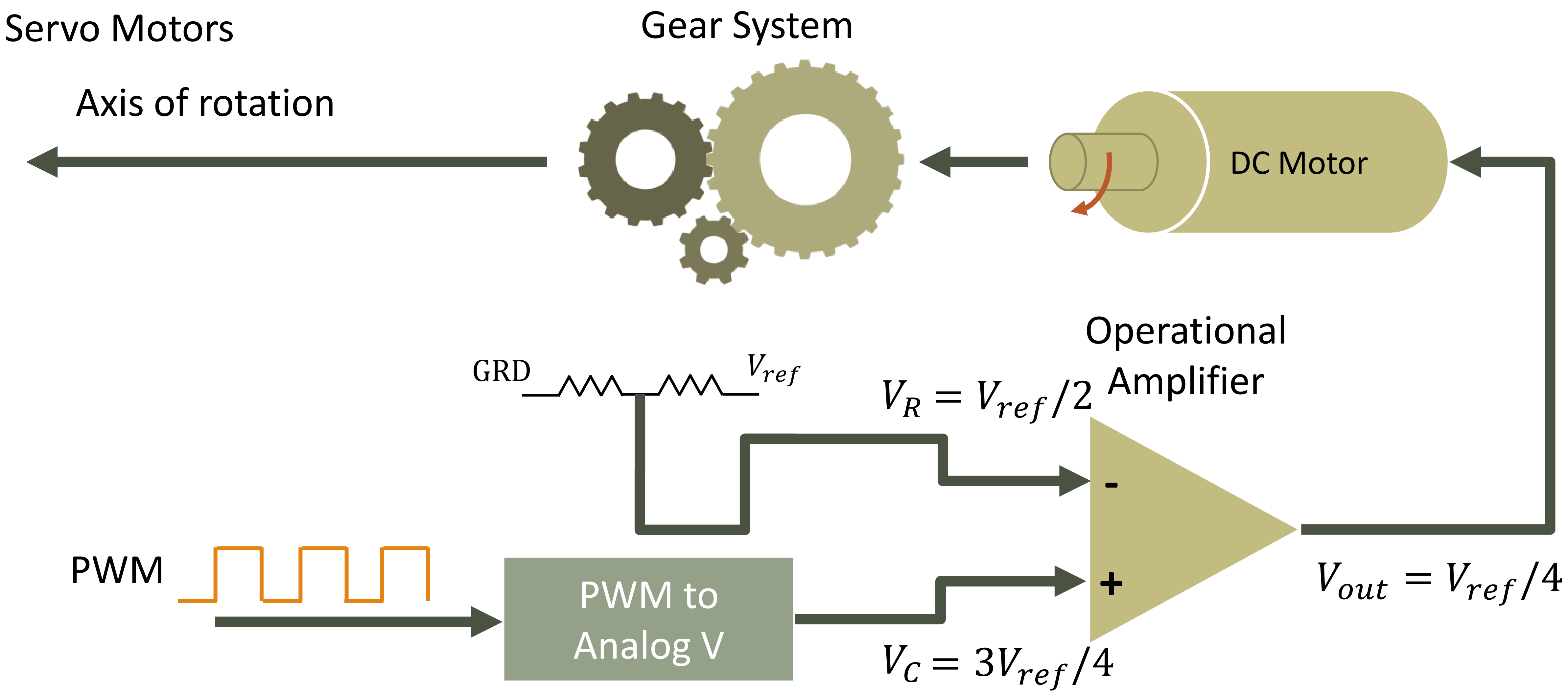


Actuator Control

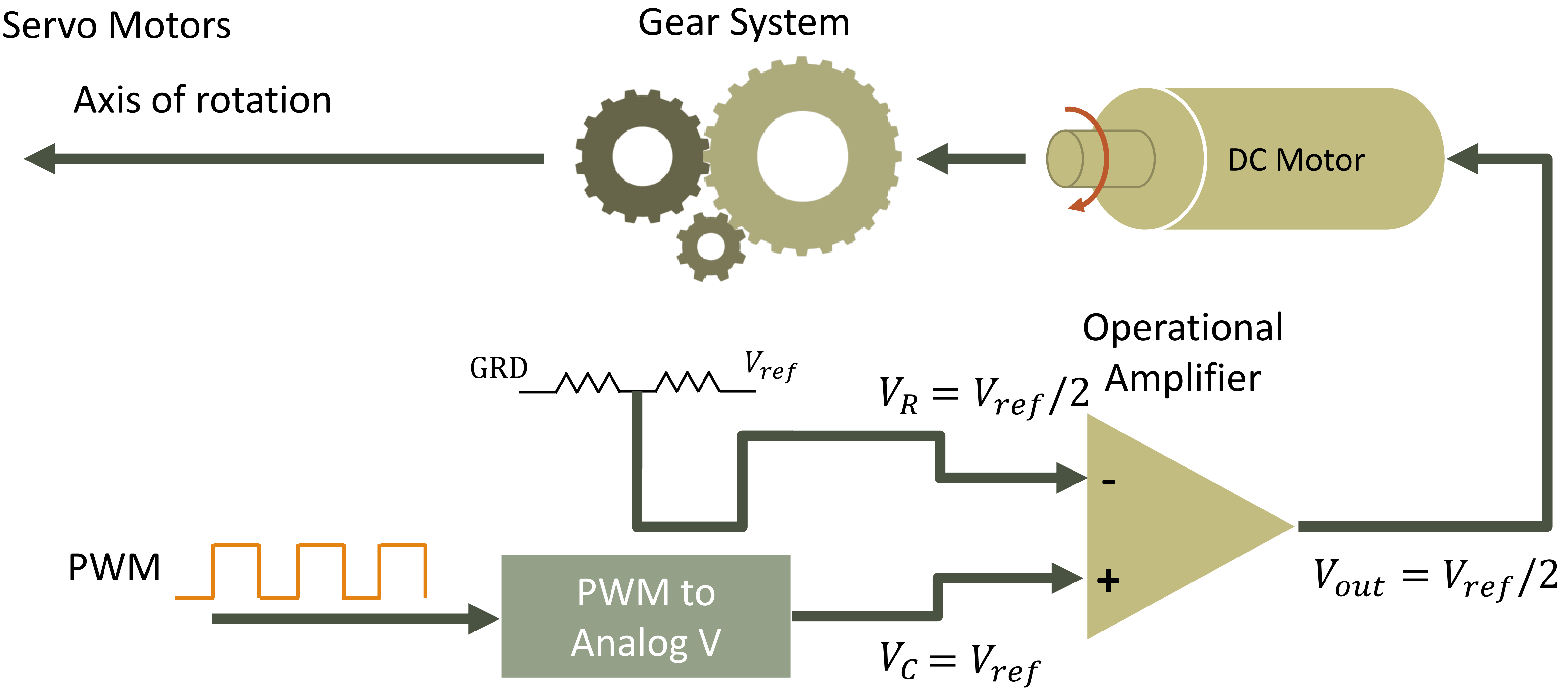
Servo Motors



Actuator Control



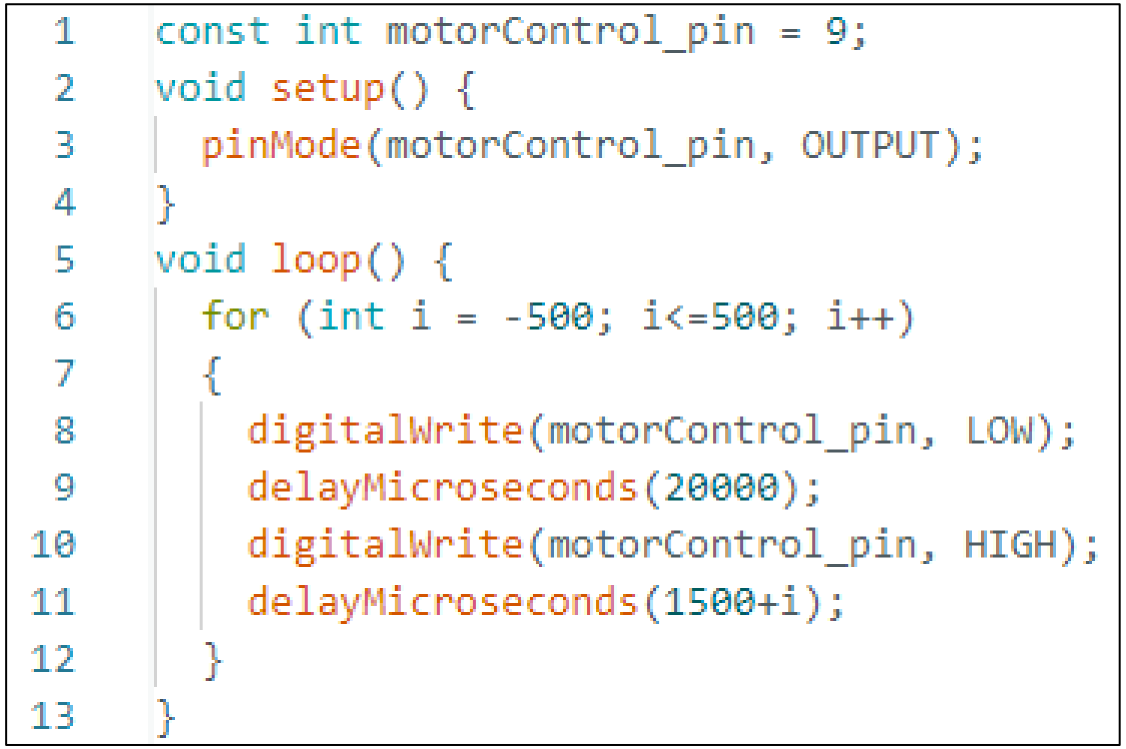
Actuator Control



Actuator Control

Servo Motors

- Example of generating a control signal via Arduino
- LOW state for 20ms
- HIGH state for 1ms to 2ms

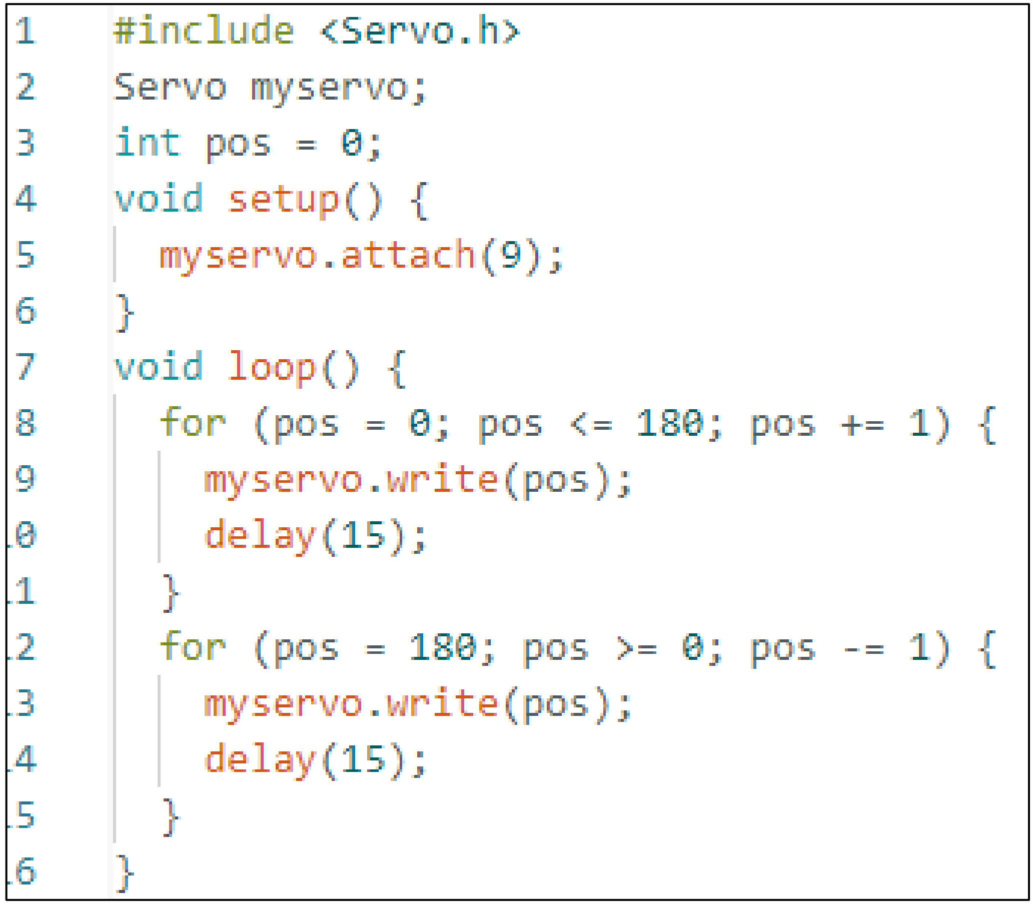


```
const int motorControl_pin = 9;
void setup() {
  pinMode(motorControl_pin, OUTPUT);
}
void loop() {
  for (int i = -500; i<=500; i++)
  {
    digitalWrite(motorControl_pin, LOW);
    delayMicroseconds(20000);
    digitalWrite(motorControl_pin, HIGH);
    delayMicroseconds(1500+i);
  }
}
```

Actuator Control

Servo Motors

- Servo.h Library
  - attach()
  - write()
  - writeMicroseconds()
  - read()
  - attached()
  - detach()



```
#include <Servo.h>
Servo myservo;
int pos = 0;
void setup() {
  myservo.attach(9);
}
void loop() {
  for (pos = 0; pos <= 180; pos += 1) {
    myservo.write(pos);
    delay(15);
  }
  for (pos = 180; pos >= 0; pos -= 1) {
    myservo.write(pos);
    delay(15);
  }
}
```

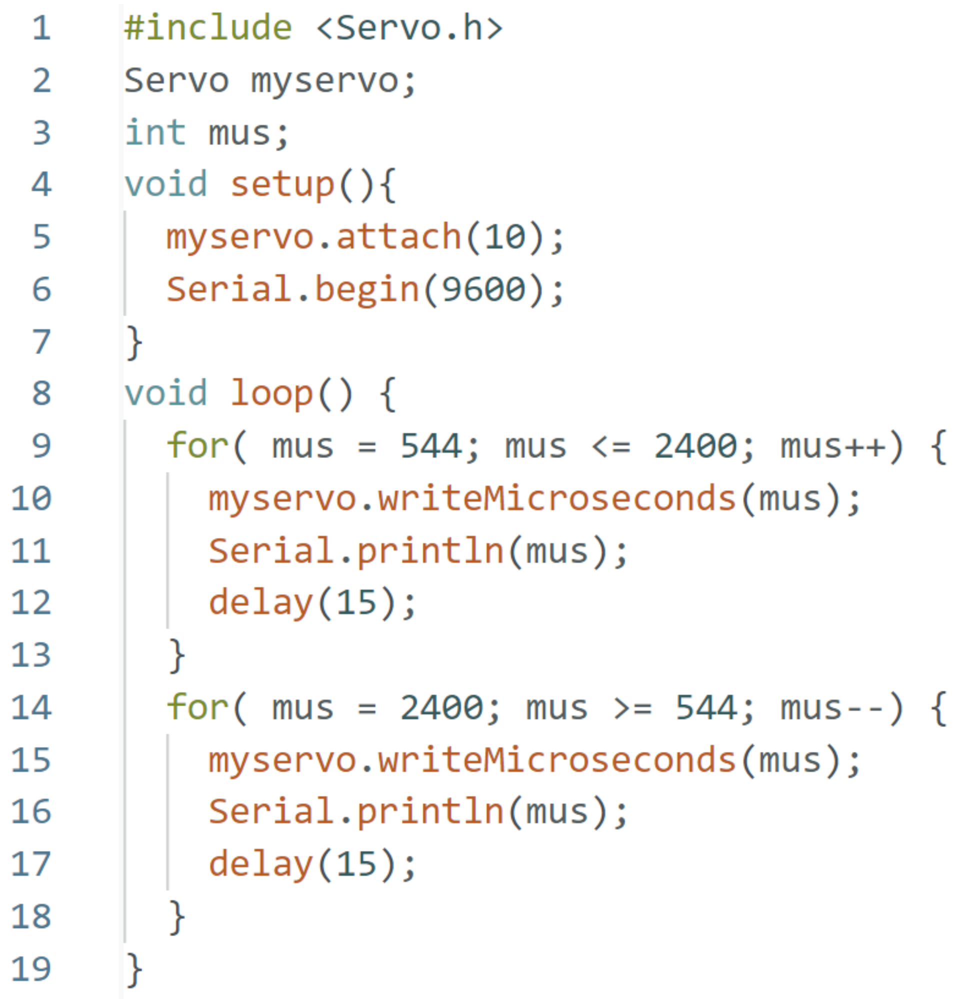
Actuator Control

Servo Motors

attach(pin_num) / **attach(pin_num, min, max)** function:

- **pin_number**: The number of control pin
  - The pins that are capable of controlling a servo motor for the Arduino Uno are 9 and 10.
- **min**: The minimum operating pulse width of the servo motor in µs.
- **max**: The maximum operating pulse width of the servo motor in µs.

Example :
myservo.attach(10): the **myservo** object controls the servo motor that is connected in pin **10**.



```
#include <Servo.h>
Servo myservo;
int mus;
void setup(){
  myservo.attach(10);
  Serial.begin(9600);
}
void loop() {
  for( mus = 544; mus <= 2400; mus++) {
    myservo.writeMicroseconds(mus);
    Serial.println(mus);
    delay(15);
  }
  for( mus = 2400; mus >= 544; mus--) {
    myservo.writeMicroseconds(mus);
    Serial.println(mus);
    delay(15);
  }
}
```

Actuator Control

Servo Motors

writeMicroseconds (val) function:

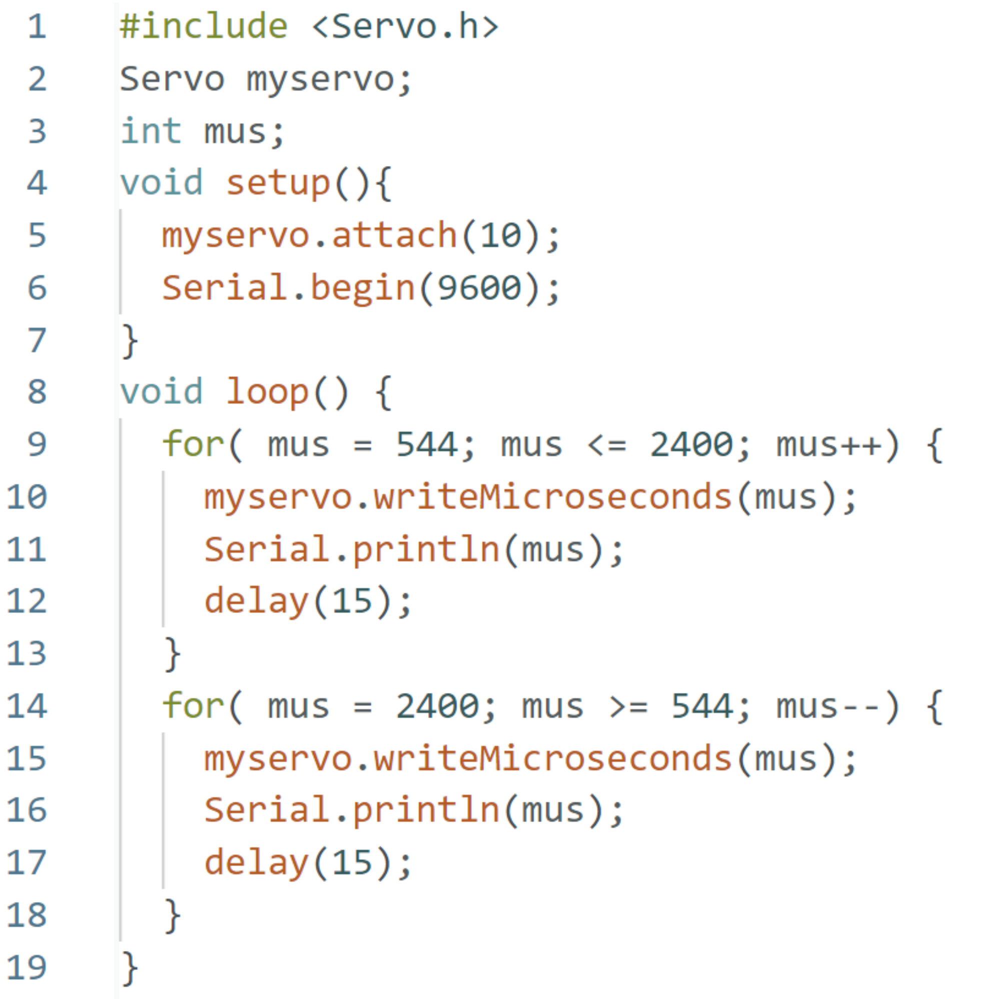
- **val**: sets the duration of generated pulses in µs

Examples:

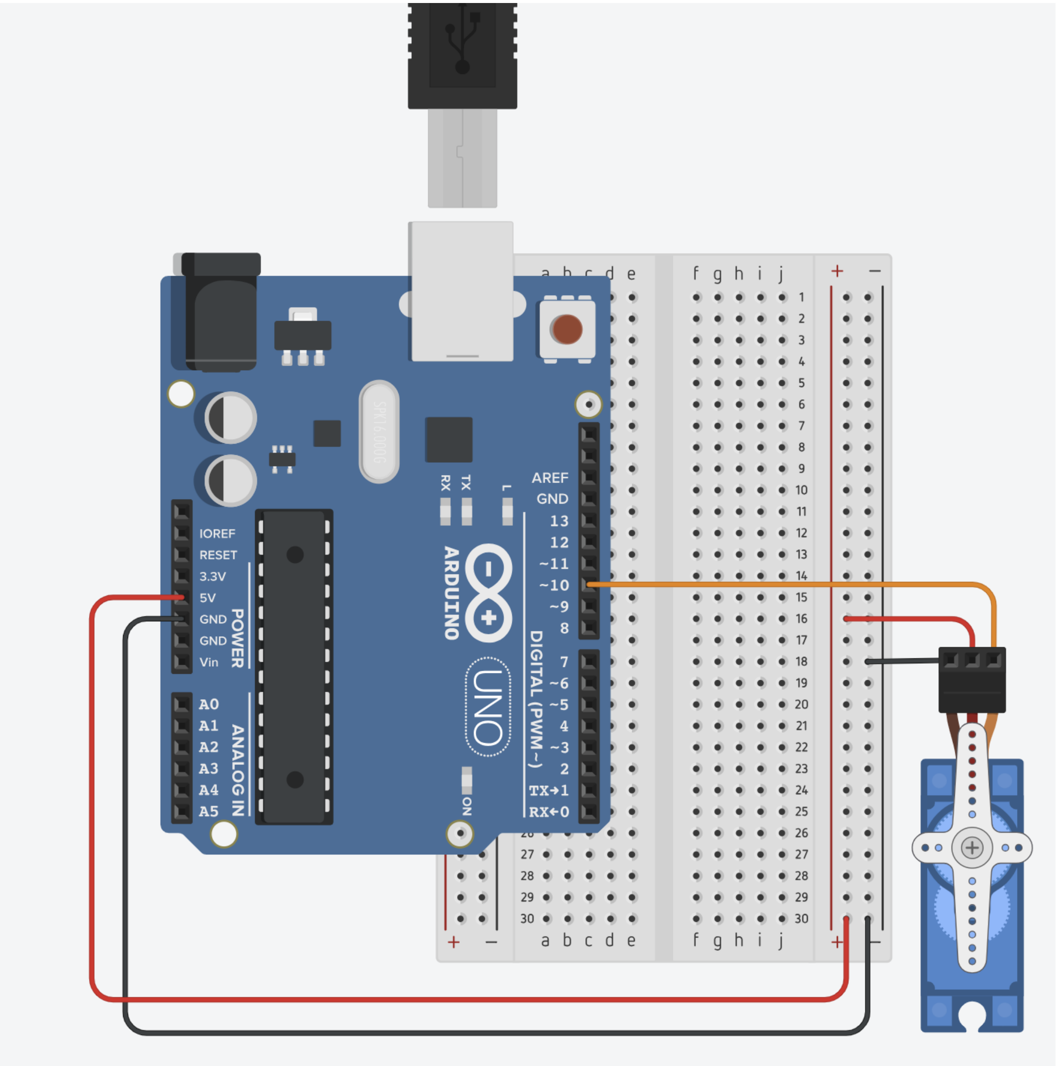
- **myservo.writeMicroseconds(1000)**: Pulse duration 1ms
- **myservo.writeMicroseconds(1500)**: Pulse duration 1.5ms
- **myservo.writeMicroseconds(2000)**: Pulse duration 2ms

Actuator Control

Servo Motors:: Arduino Example for typical servo

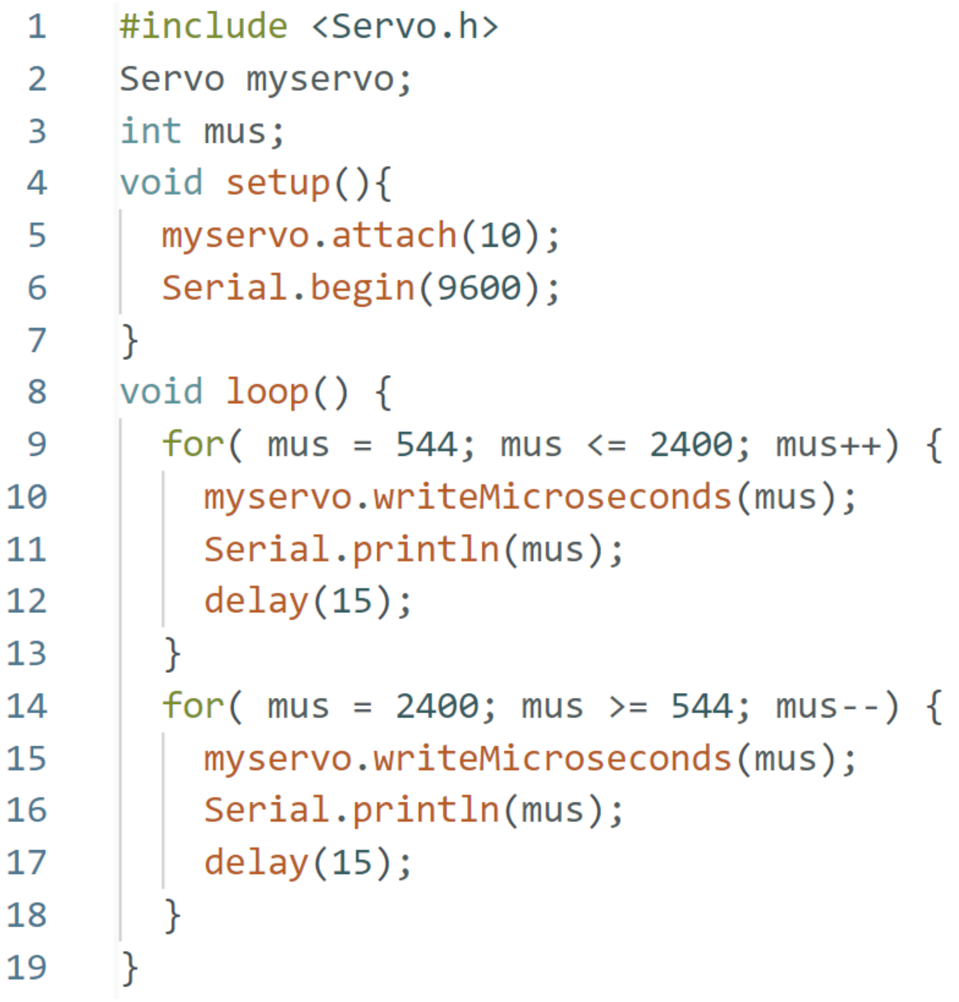


```
#include <Servo.h>
Servo myservo;
int mus;
void setup(){
  myservo.attach(10);
  Serial.begin(9600);
}
void loop() {
  for( mus = 544; mus <= 2400; mus++) {
    myservo.writeMicroseconds(mus);
    Serial.println(mus);
    delay(15);
  }
  for( mus = 2400; mus >= 544; mus--) {
    myservo.writeMicroseconds(mus);
    Serial.println(mus);
    delay(15);
  }
}
```

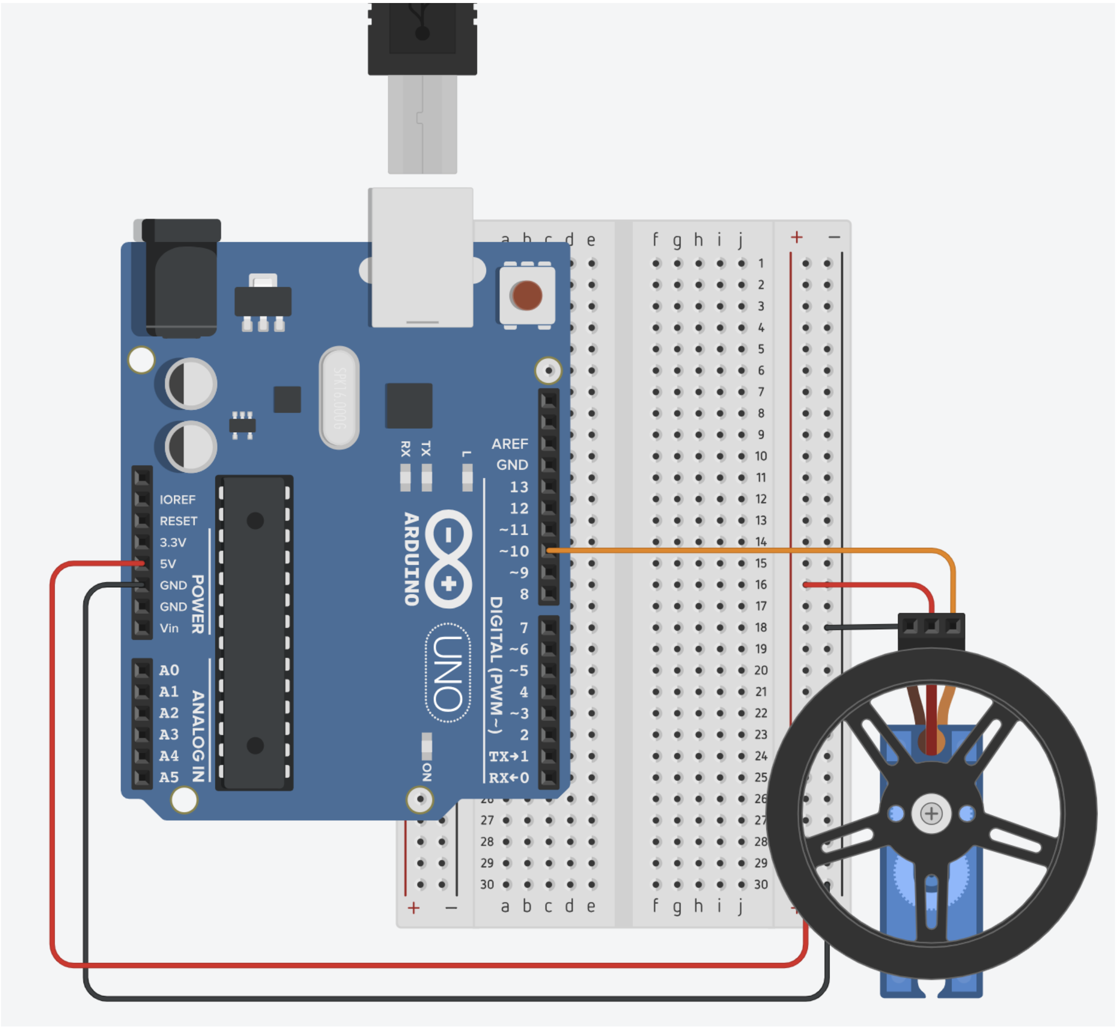


Actuator Control

Servo Motors:: Arduino Example for continuous rotation servo



```
#include <Servo.h>
Servo myservo;
int mus;
void setup(){
  myservo.attach(10);
  Serial.begin(9600);
}
void loop() {
  for( mus = 544; mus <= 2400; mus++) {
    myservo.writeMicroseconds(mus);
    Serial.println(mus);
    delay(15);
  }
  for( mus = 2400; mus >= 544; mus--) {
    myservo.writeMicroseconds(mus);
    Serial.println(mus);
    delay(15);
  }
}
```



Actuator Control

Servo Motors

write(val) function:

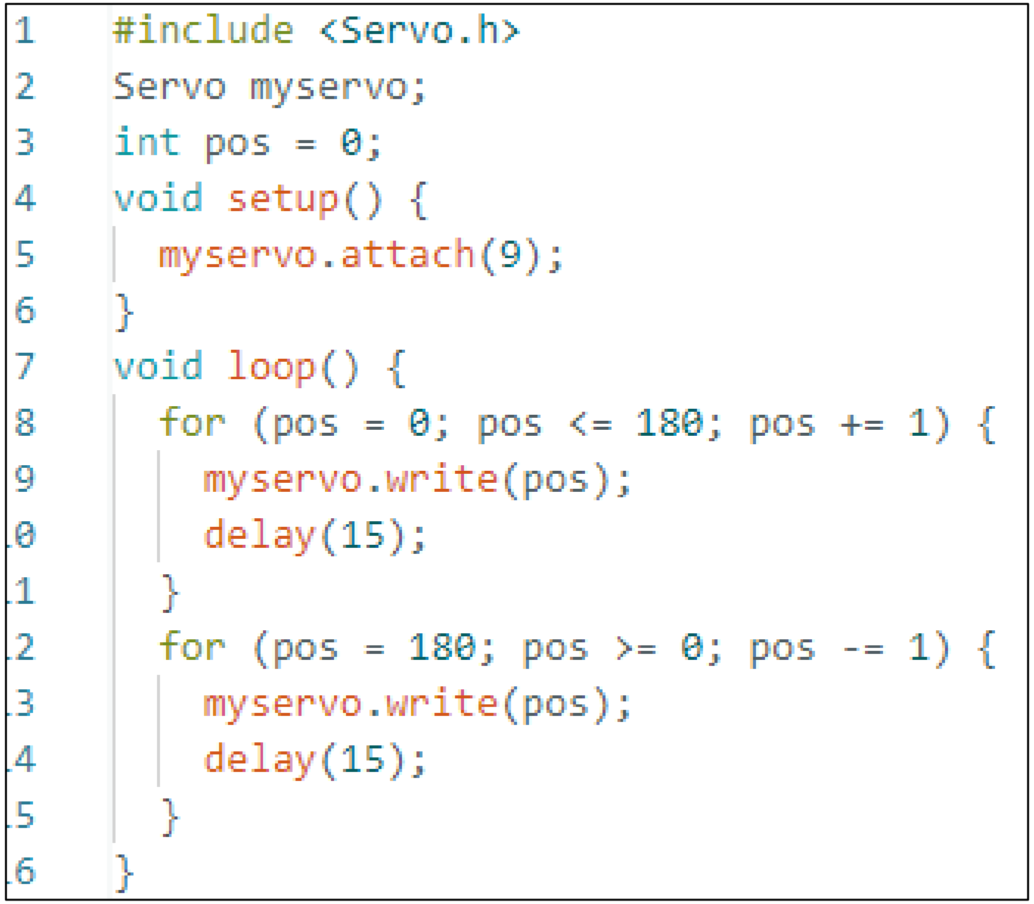
- **val**: It takes values from 0 to 180, setting the rotation state

Examples of a typical servo motor:

- **myservo.write(0)**: sets the rotation angle to 0^o
- **myservo.write(90)**: sets the rotation angle to 90^o
- **myservo.write(180)**: sets the rotation angle to 180^o

Examples of continuous rotation servo motor:

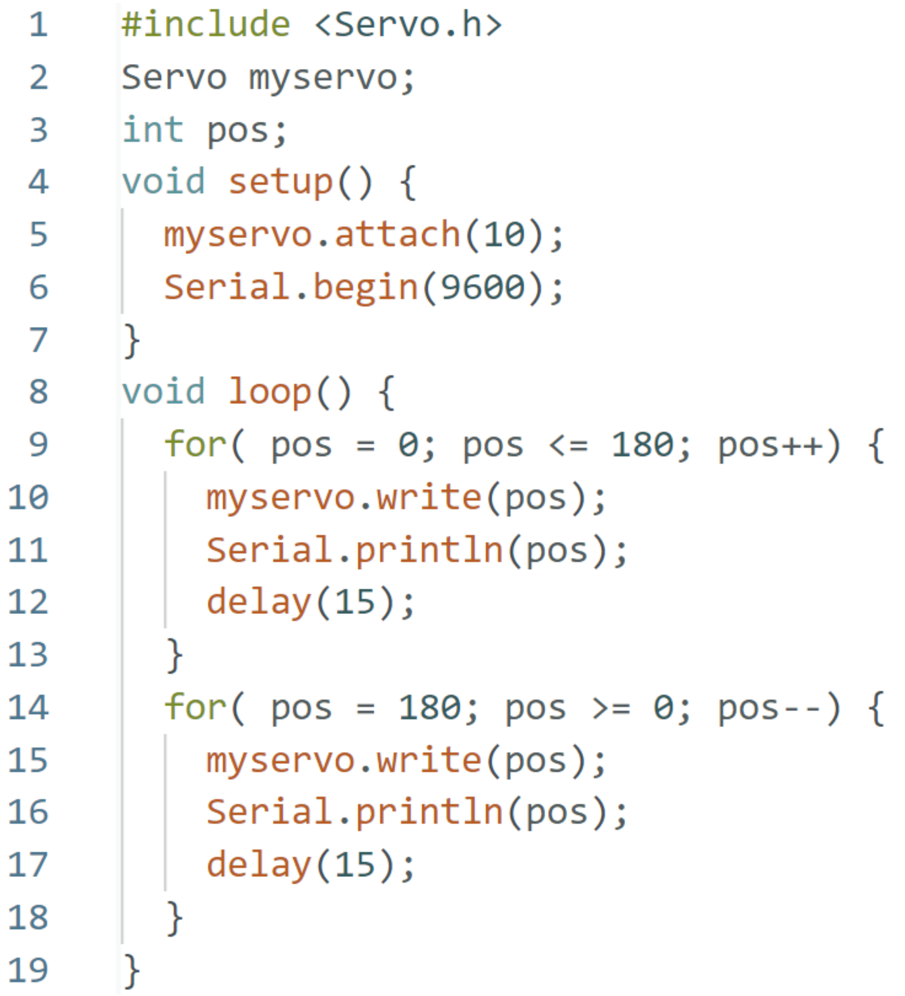
- **myservo.write(0)**: sets maximum speed counterclockwise rotation
- **myservo.write(90)**: sets zero rotation speed
- **myservo.write(180)**: sets maximum speed clockwise rotation



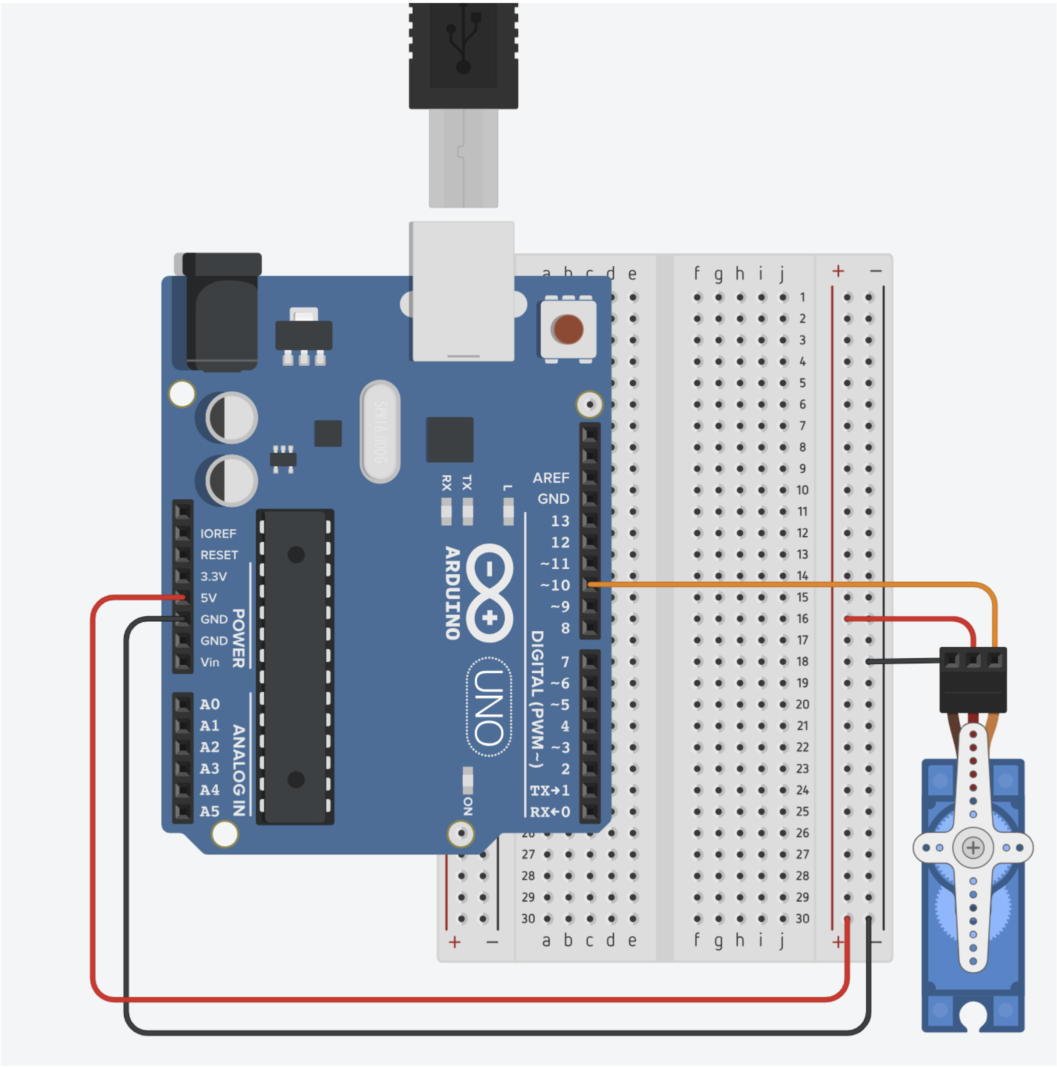
```
#include <Servo.h>
Servo myservo;
int pos = 0;
void setup() {
  myservo.attach(9);
}
void loop() {
  for (pos = 0; pos <= 180; pos += 1) {
    myservo.write(pos);
    delay(15);
  }
  for (pos = 180; pos >= 0; pos -= 1) {
    myservo.write(pos);
    delay(15);
  }
}
```

Actuator Control

Servo Motors:: Arduino Example for typical servo



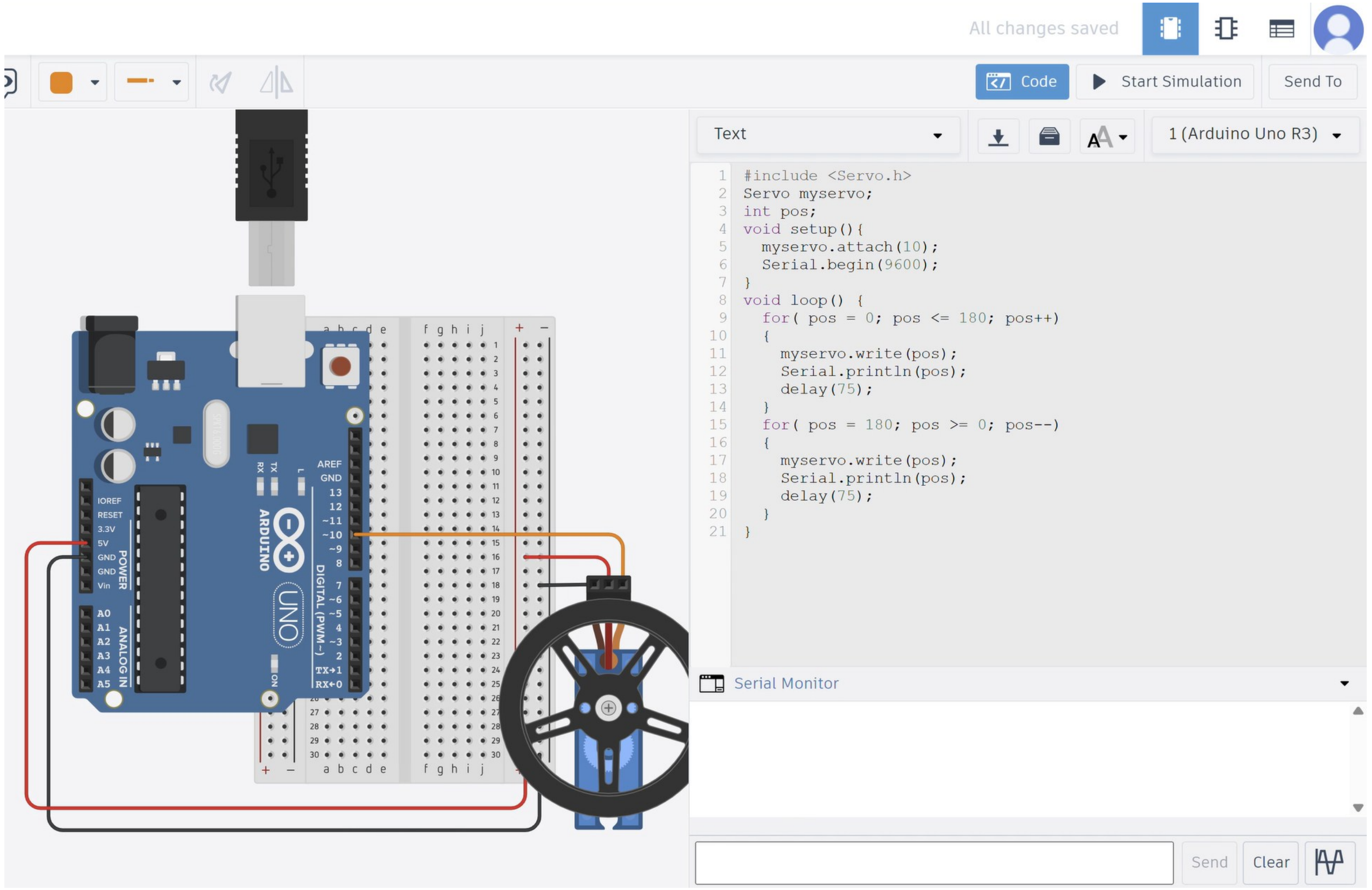
```
#include <Servo.h>
Servo myservo;
int pos;
void setup() {
  myservo.attach(10);
  Serial.begin(9600);
}
void loop() {
  for( pos = 0; pos <= 180; pos++) {
    myservo.write(pos);
    Serial.println(pos);
    delay(15);
  }
  for( pos = 180; pos >= 0; pos--) {
    myservo.write(pos);
    Serial.println(pos);
    delay(15);
  }
}
```



Actuator Control

Servo Motors:: Arduino Example for continuous rotation servo

```
#include <Servo.h>
Servo myservo;
int pos;
void setup() {
  myservo.attach(10);
  Serial.begin(9600);
}
void loop() {
  for( pos = 0; pos <= 180; pos++) {
    myservo.write(pos);
    Serial.println(pos);
    delay(15);
  }
  for( pos = 180; pos >= 0; pos--) {
    myservo.write(pos);
    Serial.println(pos);
    delay(15);
  }
}
```



```
#include <Servo.h>
Servo myservo;
int pos;
void setup(){
  myservo.attach(10);
  Serial.begin(9600);
}
void loop() {
  for( pos = 0; pos <= 180; pos++)
  {
    myservo.write(pos);
    Serial.println(pos);
    delay(75);
  }
  for( pos = 180; pos >= 0; pos--)
  {
    myservo.write(pos);
    Serial.println(pos);
    delay(75);
  }
}
```

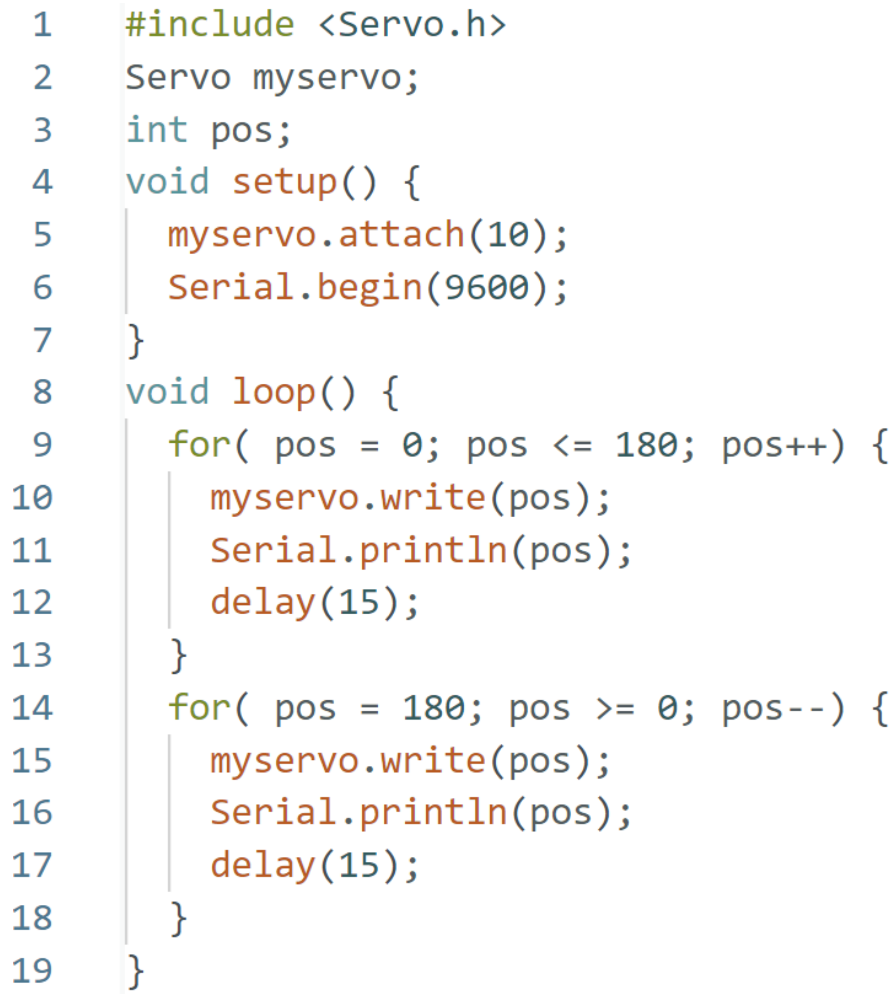
Actuator Control

Servo Motors

write(val) function:

- **Attention:** The function follows general servo motor specifications.
- The pulses have a duration of 0.544ms to 2.4ms:
  - **myservo.write(0):** Pulse duration 0.544ms
  - **myservo.write(180):** Pulse duration 2.4ms
  - step: $(2.4\text{ms} - 0.544\text{ms})/181 = 0.01025\text{ms}$

    or approximately $10\mu\text{s}$
- For those cases where the pulse duration limits are different than in the range from 0.544ms to 2.4ms, function **write(val)** can be used in the same manner.
- We need to set the correct range using the **attach(pin_num, min, max)** function.

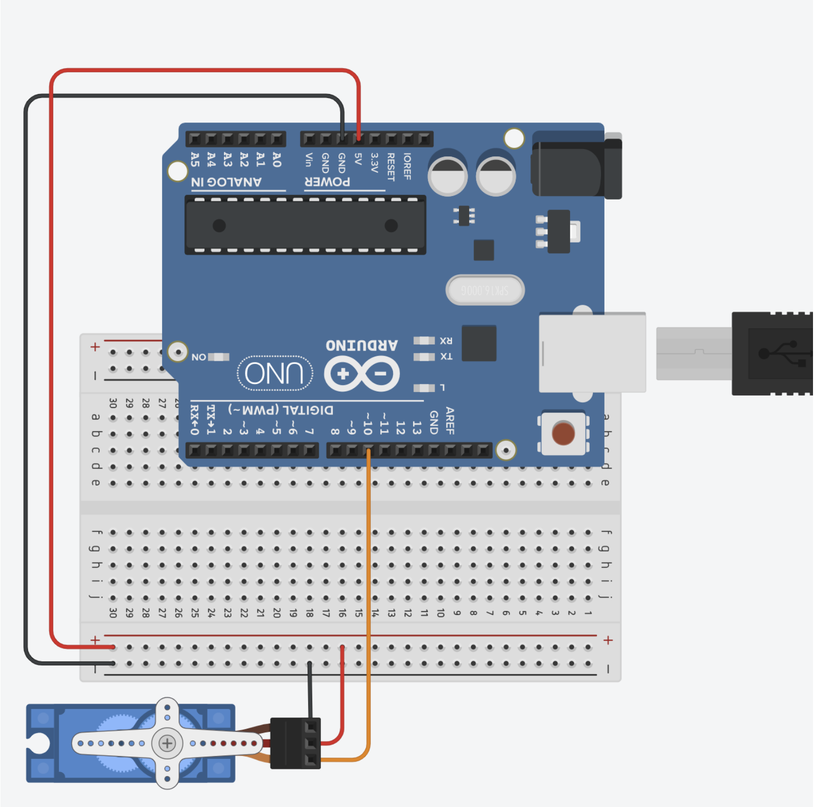


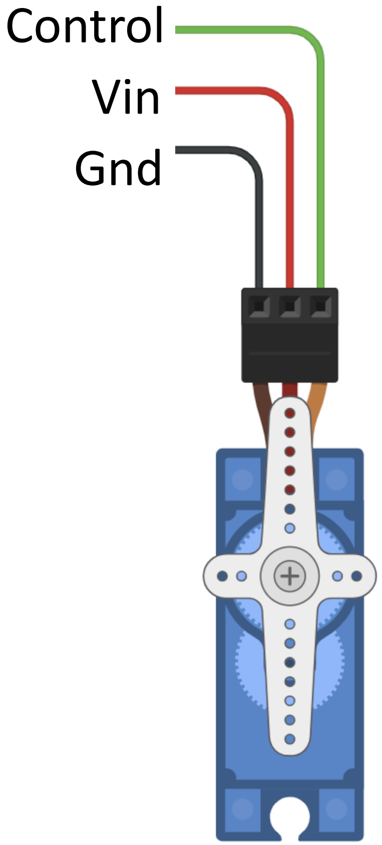
```
#include <Servo.h>
Servo myservo;
int pos;
void setup() {
  myservo.attach(10);
  Serial.begin(9600);
}
void loop() {
  for( pos = 0; pos <= 180; pos++) {
    myservo.write(pos);
    Serial.println(pos);
    delay(15);
  }
  for( pos = 180; pos >= 0; pos--) {
    myservo.write(pos);
    Serial.println(pos);
    delay(15);
  }
}
```

Actuator Control

- In the case of Servo motors, our wiring offers separate terminals for
  - Power
  - Control

- Low power Servo motor control example:

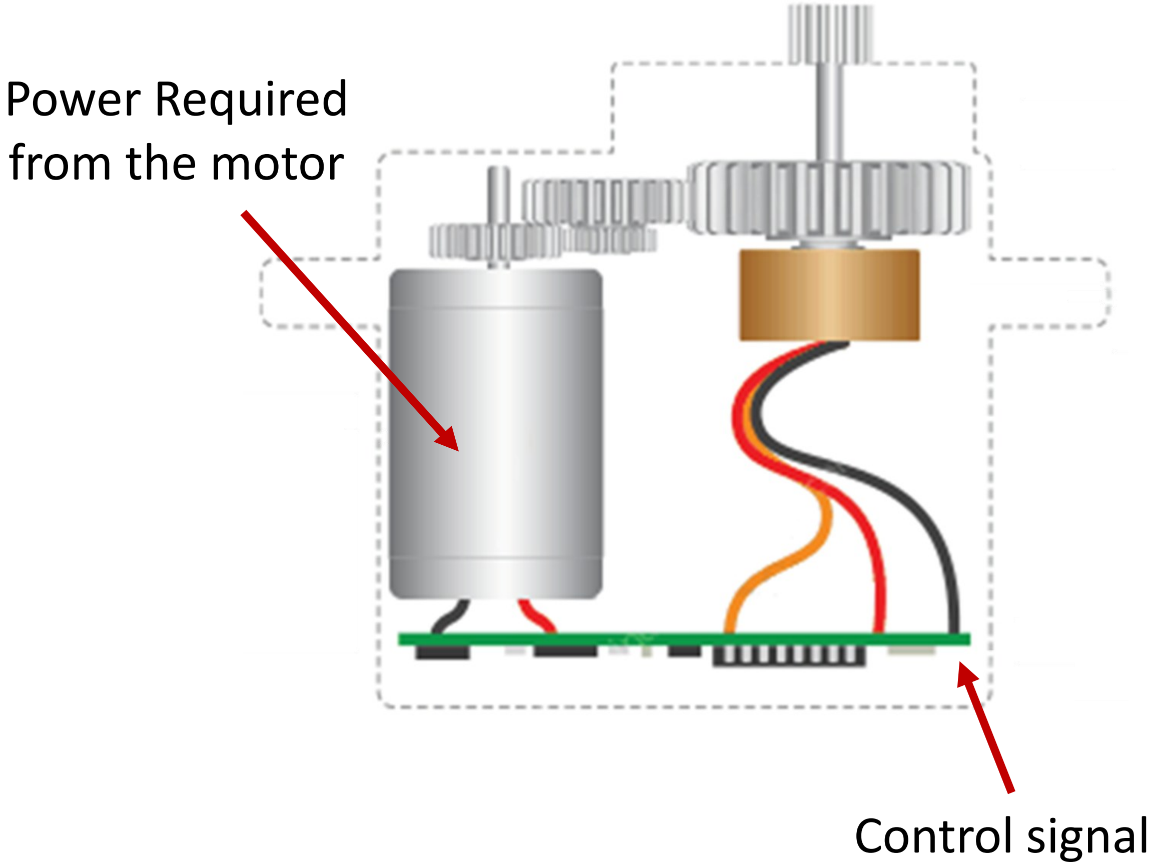




Actuator Control

- In the case of Servo motors, our wiring offers separate terminals for
  - Power
  - Control

- In many cases, the required power of the controlled loads is not covered by the capabilities of the microcontrollers.



Power Required from the motor

Control signal

Actuator Control

- Example:
  - Arduino R3: Digital pins: 20mA per output
    max: 200mA
  - Arduino R4: 8mA per output
    max: 1.2A (Vin pin) or 2A (USB)

- How many Servos of 100 mA can Arduino R3 support concurrently?

- How many Servos of 250 mA can Arduino R3 support concurrently?

- How many Servos of 1 A can Arduino R4 support concurrently?

- How many Servos of 2.1 A can Arduino R4 support concurrently?

Actuator Control

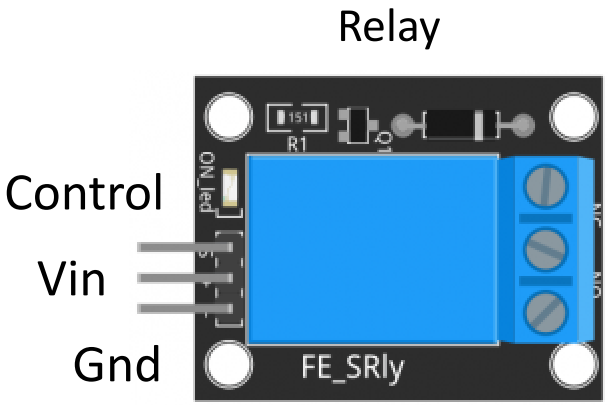
- How do I control the power supply of a 10W LED bulb through a microcontroller;

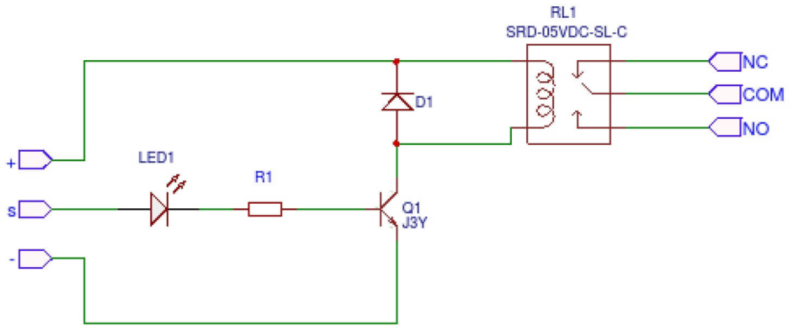
  - $V^c = ?, \ I^c_{max} = ?$
  - $V^L = ?, \ I^L_{max} = ?$

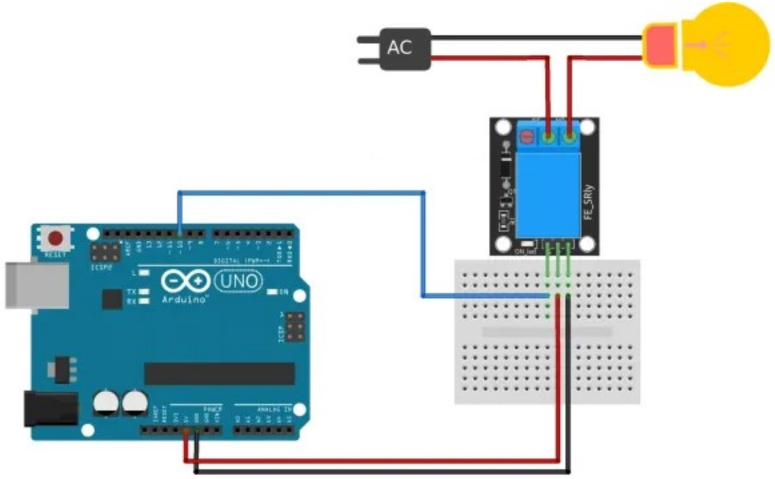
Let

  - $V^c = 5V$ DC, $I^c_{max} = 20mA$ DC
  - $V^L = 230V$ AC, $I^L_{max} = 43mA$ AC

- Solution?

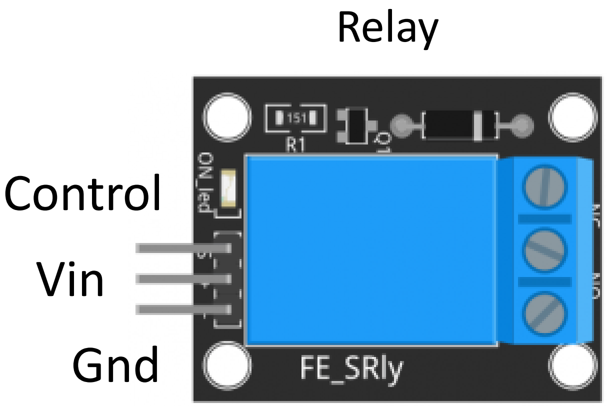


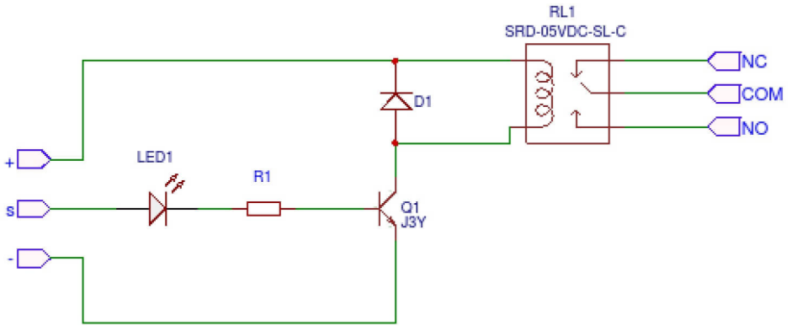


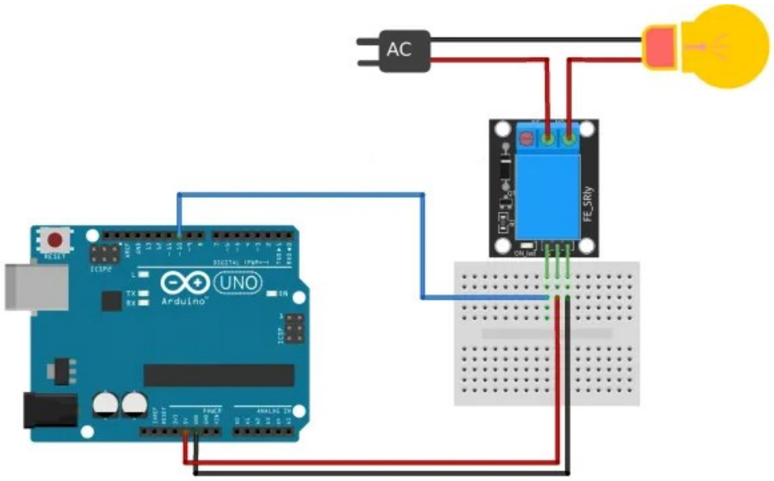


Actuator Control

- How do I control the power supply of a 10W LED bulb through a microcontroller;







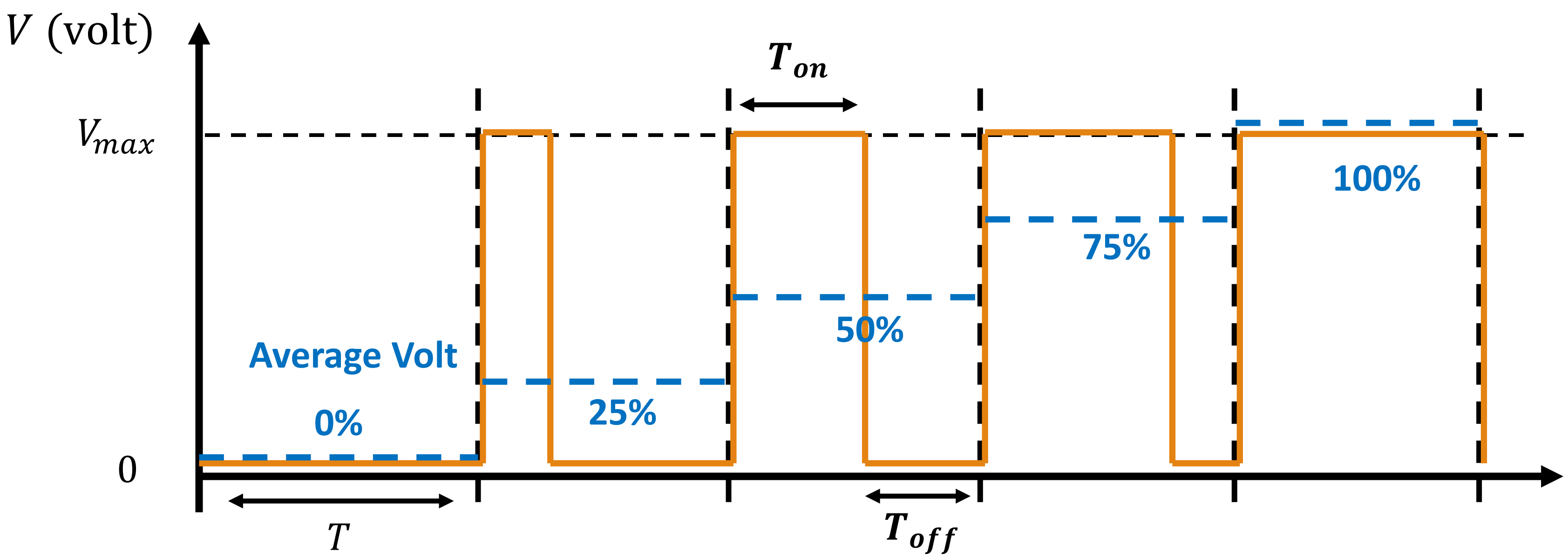
Issues:

- Depending on the case, even the relay has specific current requirements to change position on its switch.

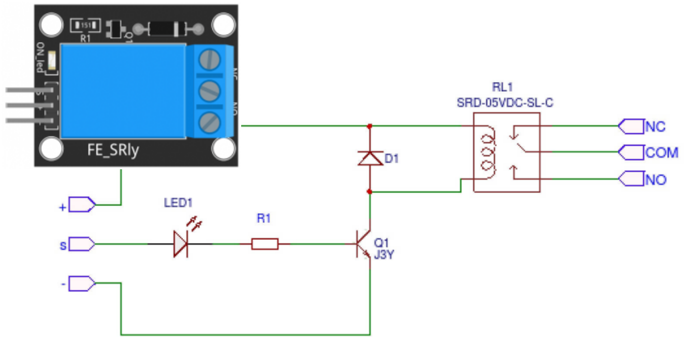
- The frequency of switching between On and Off states cannot be high due to the mechanical connection.

Actuator Control

- DC motor control via PWM



$$V_{avg} = D \cdot V_{max}$$



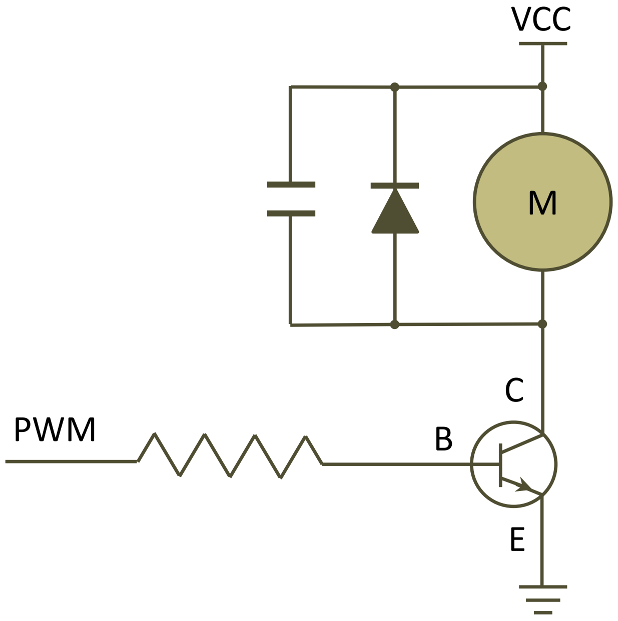
- If we could control the Relay with frequency $F = 1/T$, then we could use it.

Instead of Relay?

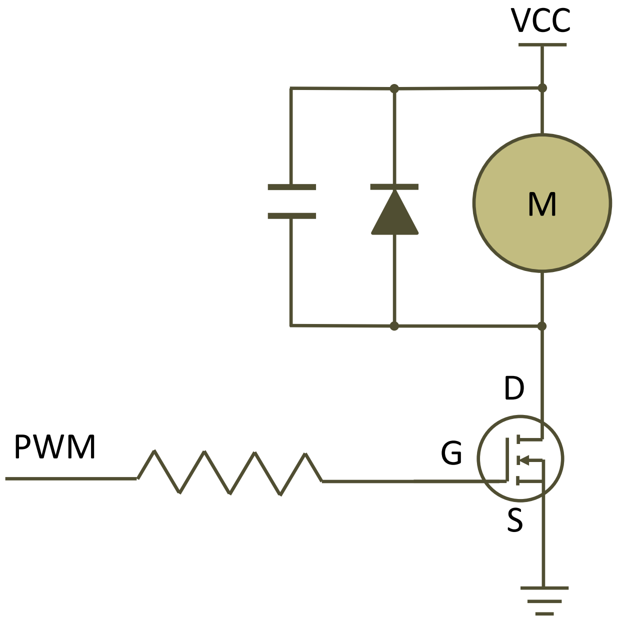
Actuator Control

- DC motor control via PWM

Transistor



Bipolar Transistor

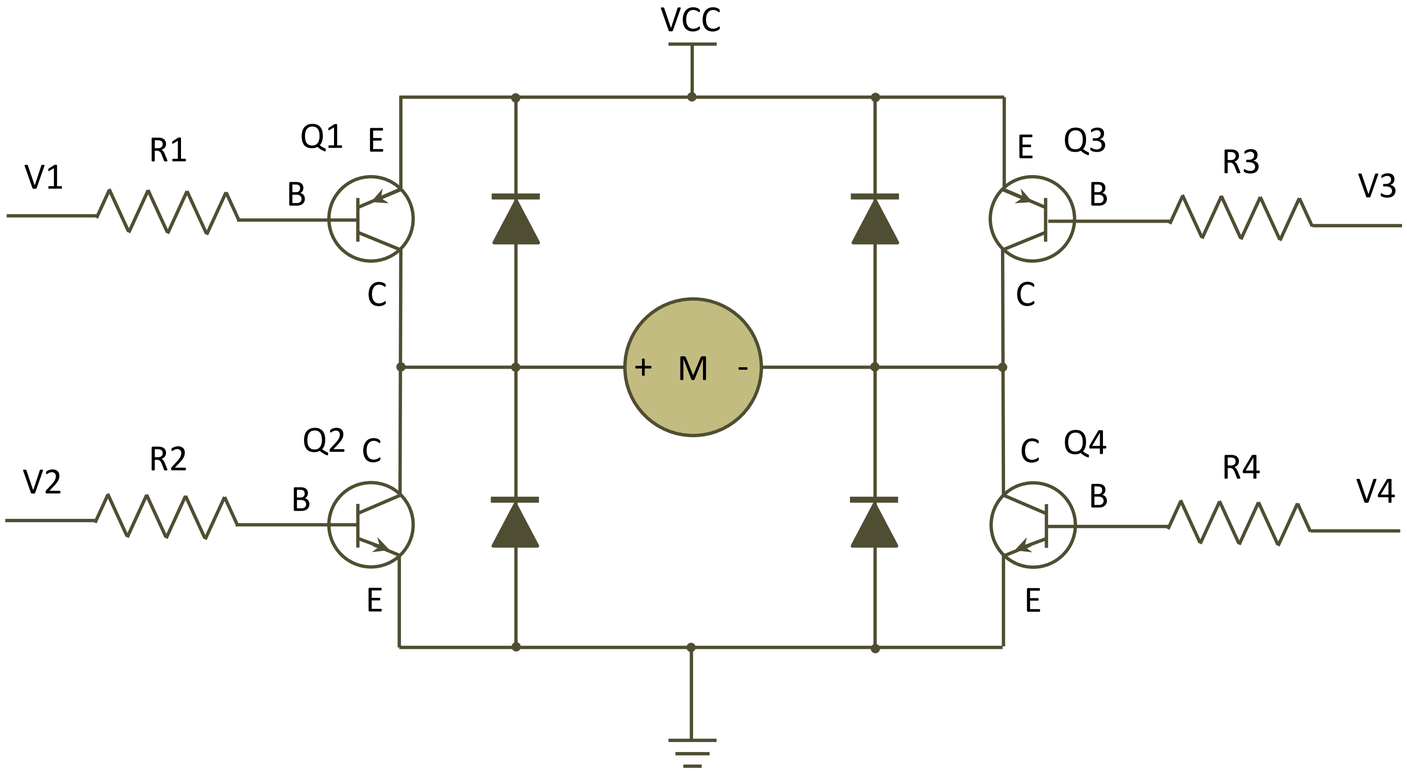


MOSFET

Actuator Control

- Bidirectional supply (rotation direction change)

H-Bridge



Actuator Control

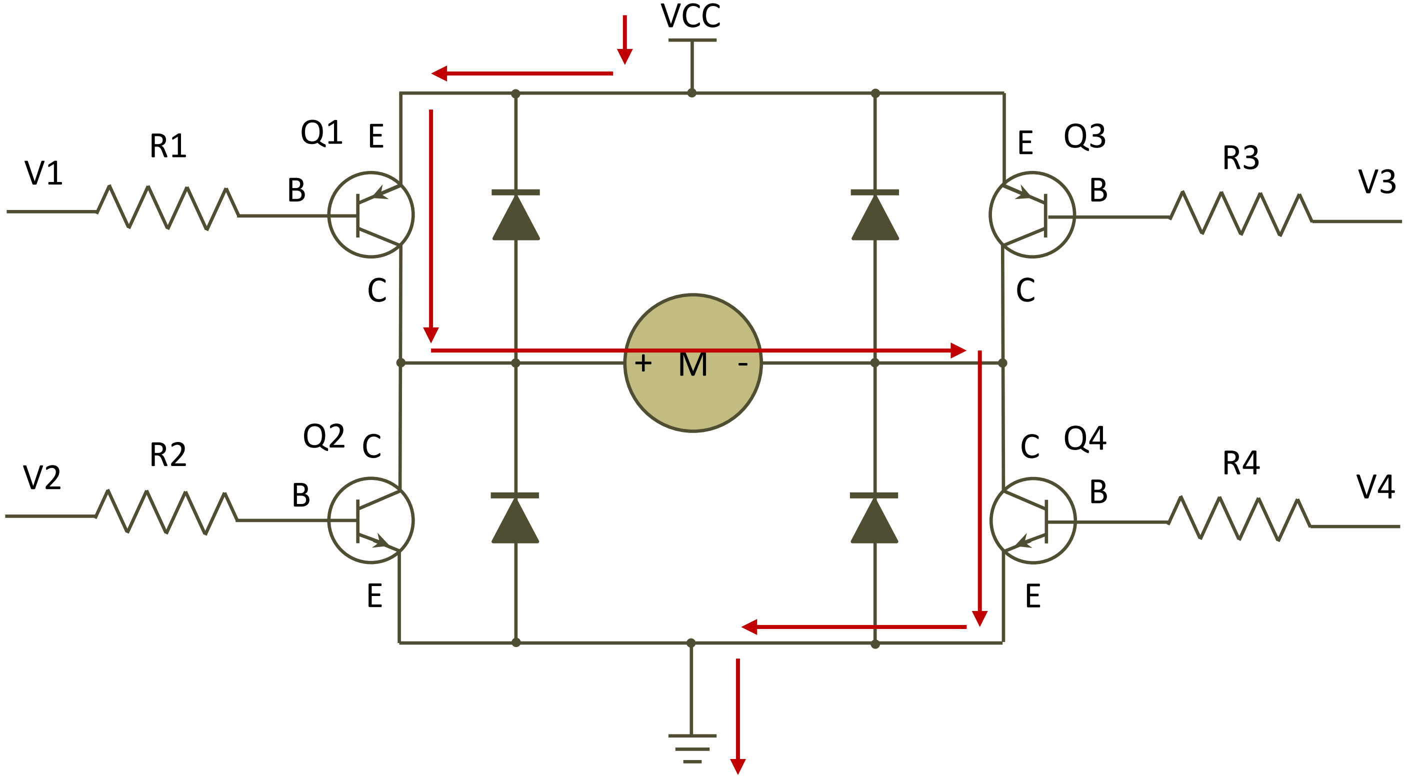
- Bidirectional supply (rotation direction change)

H-Bridge

Clockwise rotation:

- Activation: Q1, Q4

- Deactivation: Q2, Q3



Actuator Control

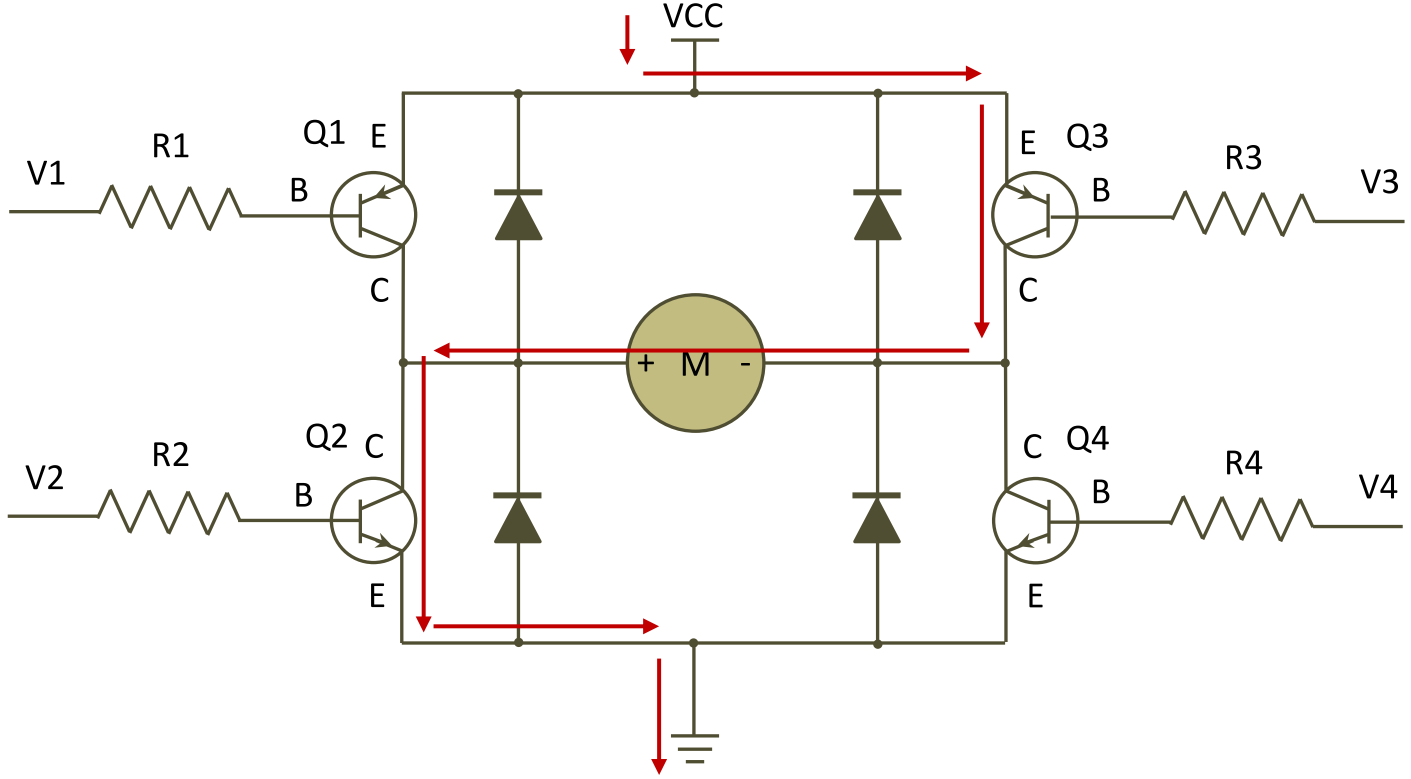
- Bidirectional supply (rotation direction change)

H-Bridge

Counterclockwise rotation:

- Activation: Q2, Q3

- Deactivation: Q1, Q4


VCC
V1
R1
Q1
E
B
C
Q3
E
B
C
R3
V3
M
+
-
V2
R2
Q2
C
B
E
Q4
C
B
E
R4
V4