

라즈베리파이 GPIO 프로그래밍

2024-1학기

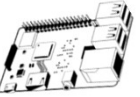
성병문

sunabove@nate.com

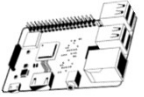
한세대학교

IT융합지능로봇공학과

저작권



1. 본 자료는 수업을 위하여 제작하고 배포합니다.
2. 본 자료의 일부나 전부를 다른 목적으로 사용을 금지합니다.



자료 표기 규칙

1. 이 강의 교재에서는 다음과 같은 표기 규칙이 사용됩니다.
 - 명령창 프롬프트를 의미합니다.
 - **[필수 파라미터 이름]**
 - [] 안은 필수적인 파라미터 이름을 나타냅니다.
 - **{ 선택 파라미터 이름 }**
 - { } 안은 선택적인 파라미터 이름을 나타냅니다.
 - **붉은 색**
 - 같은 이름으로 파일/폴더/변수 등을 관리합니다.

서보 제어

서보 소개

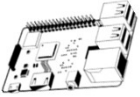
서보 제품 및 모델

서보 회로 구성

서보 캘리브레이션

서보 각도 제어 프로그래밍

서보 (Servo)



1. Servo는 전기 신호를 통해 정밀한 위치 제어가 가능한 기계 장치입니다.
 - Servo는 최대 180도까지 회전할 수 있는 제어 회로가 소형 모터가 내장되어 있습니다.
 - 빠른 속도로 GPIO 핀을 켜고 끄는 방식으로 서보를 제어할 수 있습니다.
 - PWM의 펄스 폭은 서보가 가리키는 방향을 제어합니다.
2. Servo는 다음과 같은 장점이 있습니다:
 - 정밀한 위치 제어: 작은 각도까지 정확하게 제어할 수 있습니다.
 - 간편한 사용: 제어 신호만으로 쉽게 제어할 수 있습니다.
 - 높은 토크: 작은 크기에서도 높은 토크를 제공합니다.
3. Servo의 대표적인 응용 분야는 다음과 같습니다:
 - 로봇 공학: 로봇의 관절과 같은 부분의 정밀한 움직임을 제어합니다.
 - RC 모델: RC 헬리콥터, 자동차, 비행기 등의 조향, 엘리베이터, 리더 제어에 사용됩니다.
 - 자동화 장비: 공장 자동화 장비에서 제품의 위치를 제어하거나 공정을 자동화하는 데 사용됩니다.



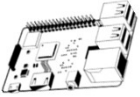
소형 서보 제품 종류



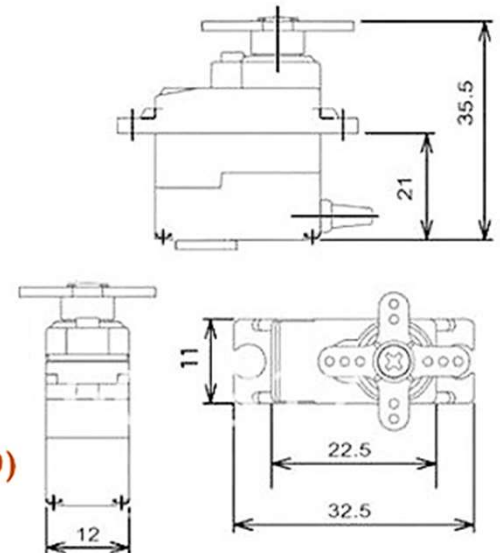
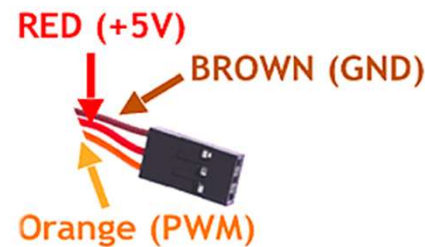
1. 교육용으로 SG90과 MG90S 모델이 많이 사용됩니다.
2. 기어 재질:
 - SG90 : 플라스틱 기어를 사용합니다.
 - 플라스틱 기어는 가볍고 저렴하지만 내구성이 상대적으로 낮아 강한 힘을 필요로 하는 응용에는 부적합할 수 있습니다.
 - MG90S : 금속 기어를 사용합니다.
 - 금속 기어는 플라스틱 기어에 비해 강도와 내구성이 높아 더 큰 힘을 견딜 수 있습니다.
 - 따라서 더 높은 토크와 내구성이 필요한 응용에 적합합니다.
3. 토크:
 - SG90 : 약 1.8 kg·cm (4.8V 기준)의 토크를 제공합니다.
 - MG90S : 약 2.2 kg·cm (4.8V 기준)의 토크를 제공합니다.
4. 무게:
 - SG90 : 약 9g
 - MG90S : 약 13.4g.
5. 크기:
 - 두 모델 모두 크기는 유사하며, 23mm x 12.2mm x 29mm 정도의 크기를 가집니다.

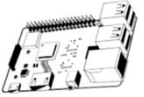


서보 MG90S 모델



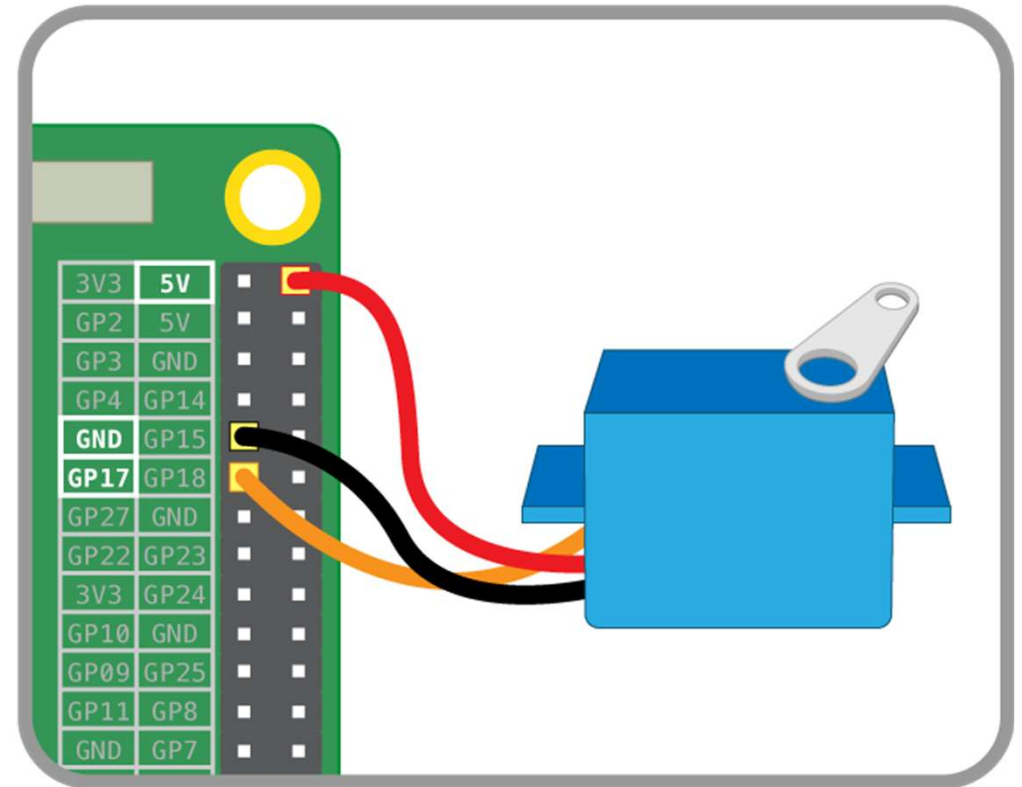
1. 높은 토크:
 - 약 $2.2 \text{ kg}\cdot\text{cm}$ (4.8V 기준)의 토크
 - 소형 서보 모터 중에서 비교적 높음
2. 정밀한 제어:
 - 0.08초/60도(4.8V 기준)의 빠른 응답 속도
3. 작고 가벼운 디자인:
 - 약 13.4g의 무게
4. 넓은 작동 전압 범위:
 - 4.8V ~ 6.0V



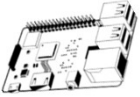


서보 회로 구성

1. 서보는 3개의 핀을 제공
 - 접지선: 갈색/검은색
 - 전원선: 빨간색
 - 신호선 : 노란색/주황색
2. 접지선을 GND 핀에 연결
3. 전원선을 5V 핀에 연결
4. 신호선을 GPIO 17번 핀에 연결



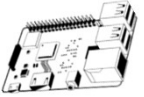
캘리브레이션(Calibration) 목적



1. 캘리브레이션은 센서나 장치가 정확하게 작동하도록 기준 값에 맞춰 조정하고 확인하는 과정입니다.
2. 캘리브레이션은 측정 장치나 제어 시스템의 성능을 보장하고 신뢰할 수 있는 데이터를 제공하기 위해 필수적인 단계입니다.
3. 목적:
 - 정확도 향상: 장치의 측정값이나 출력값의 정확성을 높입니다.
 - 신뢰성 보장: 장치가 일관되고 신뢰할 수 있는 데이터를 제공하도록 합니다.
 - 성능 유지: 장치가 최상의 성능을 유지하도록 정기적으로 점검하고 조정합니다.
 - 규정 준수: 산업 표준이나 법적 규제를 준수하기 위하여 필요할 수 있습니다.

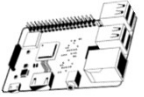


캘리브레이션(Calibration) 과정



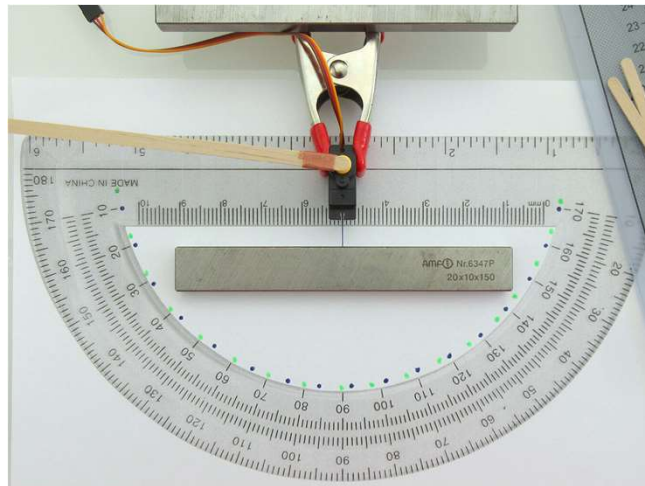
1. 기준 장치 준비:
 - 캘리브레이션을 위해 신뢰할 수 있는 기준 장치나 표준을 준비합니다.
 - 기준 장치는 매우 높은 정확성을 갖춘 장치로, 측정값의 참조로 사용됩니다.
2. 초기 측정값 확인:
 - 캘리브레이션 할 장치의 초기 측정값을 기록합니다.
 - 이 값은 기준 장치의 값과 비교됩니다.
3. 비교 및 조정:
 - 장치의 측정값과 기준 장치의 값을 비교합니다.
 - 차이가 있을 경우 장치의 설정을 조정하여 기준 값에 맞추도록 합니다.
4. 재 측정 및 확인:
 - 조정 후 다시 측정하여 장치의 측정값이 기준 값과 일치하는지 확인합니다.
 - 필요시 추가 조정을 합니다.
5. 기록 및 문서화:
 - 캘리브레이션 결과를 기록하고 문서화합니다.
 - 이는 추후 검증이나 유지보수를 위해 필요합니다.



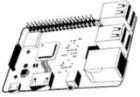


Servo Calibration

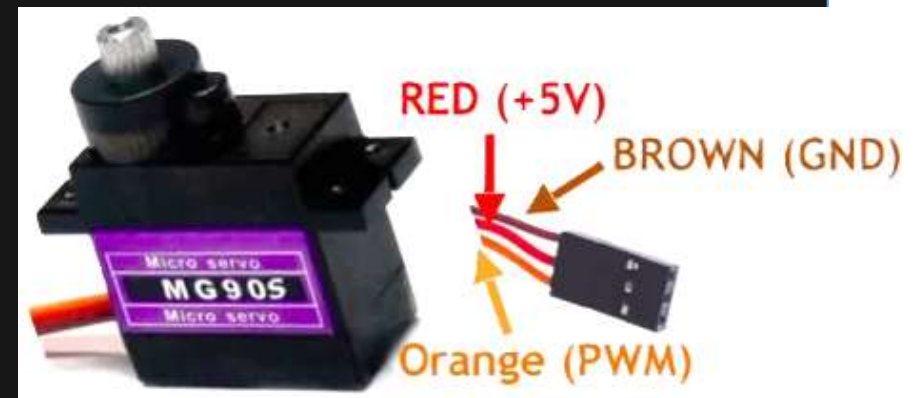
1. 서보 캘리브레이션(Servo Calibration)은 서보의 정확한 위치 제어를 위하여 듀티 사이클 값과 서보 모터의 실제 각도를 맞추는 과정입니다.
2. 서보 모터마다 약간의 차이가 있을 수 있으므로, 캘리브레이션으로 서보 모터에 대한 최적의 듀티 사이클 값을 찾는 것이 중요합니다.



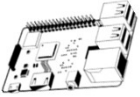
Servo 캘리브레이션



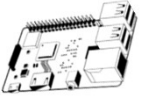
```
1. # servo_cali.py
2. # 서보를 중앙에 위치시킵니다.
3. # 서보 암을 중앙 방향으로 부착합니다.
4. from gpiozero import Servo
5. from time import sleep
6. servo = Servo(17)
7. print( "Start calibrating ..." )
8. for _ in range( 5 ) :
9.     servo.mid()
10.    sleep( 2 )
11. pass
12. print( "Done calibrating." )
```



Servo 동작 테스트

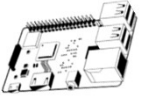


```
1.  # servo_test.py # 서보 암을 최소 -> 중간 -> 최대 -> 중간 설정을 무한 반복 합니다.
2.  from gpiozero import Servo
3.  from time import sleep
4.  servo = Servo(17)
5.  while True:
6.      servo.min()
7.      print( "servo min")
8.      sleep( 2 )
9.      servo.mid()
10.     print( "servo mid")
11.     sleep( 2 )
12.     servo.max()
13.     print( "servo max")
14.     sleep( 2 )
15.     servo.mid()
16.     print( "servo mid")
17.     sleep( 2 )
```



Servo 각도 value 제어

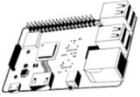
```
1.  # servo_value.py
2.  # 서보의 위치를 값을 통하여 제어합니다.
3.  from gpiozero import Servo
4.  from time import sleep
5.  servo = Servo(17)
6.  value = -1.0 ; dir = 1
7.  while True :
8.      value += (dir*0.1)
9.      if value > 1 :
10.         dir = -1
11.         value = 1.0
12.     elif value < -1 :
13.         dir = +1
14.         value = -1.0
15.     pass
16.     servo.value = value
17.     sleep( 2 )
18.     print( f"value = {value:+1.1f}, servo.value = {servo.value:+1.1f}, dir = {dir}" )
19.     pass
```



Servo 각도 angle 제어

```
1.  # servo_angle.py  # 서보의 위치를 값을 통하여 제어합니다.
2.  from gpiozero import AngularServo
3.  from time import sleep
4.  servo = AngularServo(17)
5.  angle = -90 ; dir = 1
6.  while True :
7.      angle += (dir*5)
8.      if angle >= 90 :
9.          dir = -1
10.         angle = 90
11.     elif angle <= -90 :
12.         dir = +1
13.         value = -90
14.     pass
15.     servo.angle = angle
16.     sleep( 2 )
17.     print( f"angle = {angle:+}, servo.angle = {servo.angle:+}, dir = {dir:+}" )
```

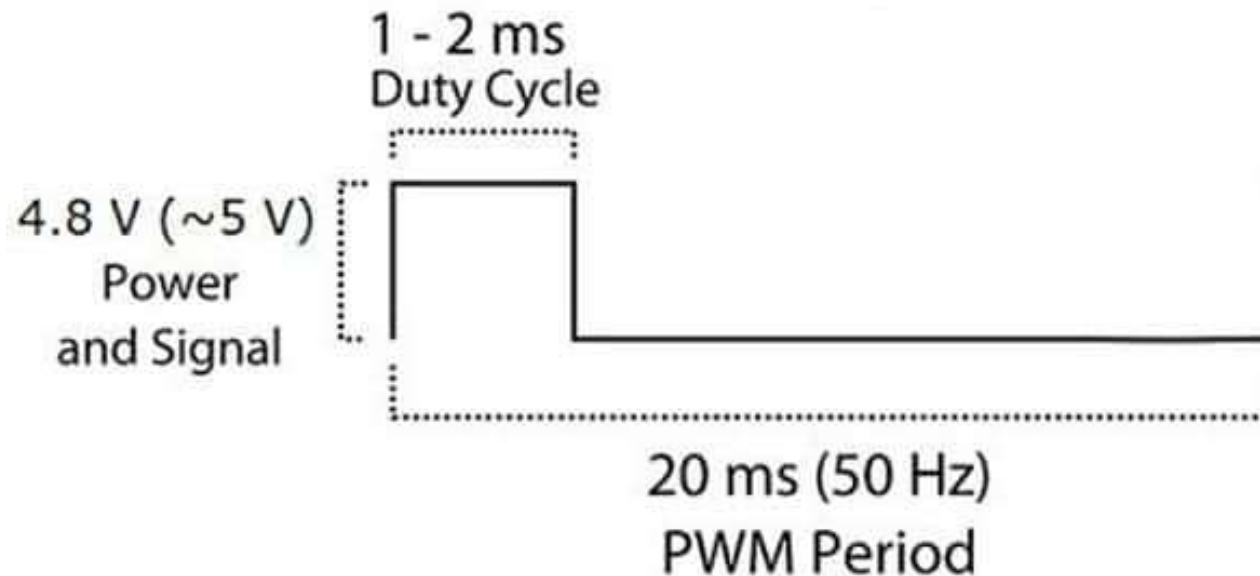
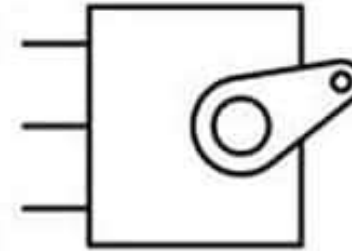
Servo PWM DutyCycle

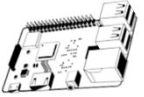


PWM=Orange (⏏)

Vcc = Red (+)

Ground=Brown (-)

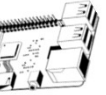




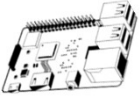
Servo 각도 PWM 제어

```
1. # servo_pwm.py
2. # GPIO 핀 17을 사용하여 PWM 신호를 생성합니다.
3. import RPi.GPIO as GPIO # 표준 Raspberry Pi GPIO 라이브러리 임포트
4. from time import sleep # 프로그램에 대기(일시 정지) 기능 추가
5. GPIO.setmode(GPIO.BCM) # 핀 번호 체계를 BCM 모드로 설정
6. GPIO.setup(17, GPIO.OUT) # 핀 17을 출력으로 설정
7. p = GPIO.PWM(17, 50) # 핀 17을 PWM 핀으로 설정하고 주파수를 50Hz로 설정
8. p.start(0) # PWM 시작, 듀티 사이클을 0으로 설정
9. sleep(1) # 1초 대기
10. for dc in range( 12 + 1 ):
11.     print( f"dc = {dc}" )
12.     p.ChangeDutyCycle( dc ) # 듀티 사이클을 변경하여 서보 모터를 특정 각도로 이동
13.     sleep( 0.5 )
14. pass
15. p.stop() # PWM을 중지
16. GPIO.cleanup() # GPIO 핀을 기본값으로 재설정
```

과제 : 서버 각도 조절



1. 아래의 기능을 무한 반복하는 기능을 구현하세요.
 - 서버의 각도를 0도에서 180도까지 약 15도씩 부드럽게 증가시킵니다.
 - 서버의 각도를 180도에서 0도까지 약 15도씩 부드럽게 감소시킵니다.
2. 제출할 결과물
 - 결과 시연 YouTube 동영상 링크
 - 동영상 파일 자체를 직접 첨부하지 마십시오.
 - 프로그램 소스 mySensorPwm.py 파일



서보 각도 조절 소스 코드

```
1. # servo_pwm_loop.py # GPIO 핀 17을 사용하여 PWM 신호로 서보의 각도를 제어합니다.
2. import Rpi.GPIO as GPIO # 표준 Raspberry Pi GPIO 라이브러리 импорт
3. from time import sleep # 프로그램에 대기(일시 정지) 기능 추가
4. GPIO.setmode(GPIO.BCM) # 핀 번호 체계를 BCM 모드로 설정
5. GPIO.setup(17, GPIO.OUT) # 핀 17을 출력으로 설정
6. p = GPIO.PWM(17, 50) # 핀 17을 PWM 핀으로 설정하고 주파수를 50Hz로 설정
7. p.start(0) # PWM 시작, 듀티 사이클을 0으로 설정
8. sleep(1) # 1초 대기
9. while True :
10.     for i in [ 1, -1 ] :
11.         for dc in range( 6 - 6*i, 6 + 6*i + i, i ):
12.             print( f"dc = {dc}" )
13.             p.ChangeDutyCycle( dc ) # 듀티 사이클을 변경하여 서보 모터를 특정 각도로 이동
14.             sleep( 0.5 )
15.         pass
16.     pass
17. pass
18. p.stop() # PWM을 중지
19. GPIO.cleanup() # GPIO 핀을 기본값으로 재설정
```

- 끝 -