programmers

자율주행 데브코스

ROS 예제코드 분석

㈜자이트론

허성민

smher@xytron.co.kr

programmers

Contents

programmers

ROS 예제코드 분석

과제 설명

파이썬 예제코드 분석

Xytron

- 2개 파일의 소스코드 분석
  - teacher.py
  - student.py

teacher.py

```
1   #!/usr/bin/env python
2
3   import rospy
4   from std_msgs.msg import String
5
6   rospy.init_node('teacher')
7
8   pub = rospy.Publisher('my_topic', String)
9
10  rate = rospy.Rate(2)
11
12  while not rospy.is_shutdown():
13      pub.publish('call me please')
14      rate.sleep()
```

student.py

```
1   #!/usr/bin/env python
2
3   import rospy
4   from std_msgs.msg import String
5
6   def callback(msg):
7       print msg.data
8
9   rospy.init_node('student')
10
11  sub = rospy.Subscriber('my_topic', String, callback)
12
13  rospy.spin()
14
```

결과물 제출 방법

- teacher.py / student.py
  - 한 줄 한 줄 최대한 자세히 코드 설명
  - 각각 어떤 일을 하는 코드인지 설명

- teacher.py / student.py
  - 2개 파일이 서로 어떻게 상호동작하는 것인지 동작 시나리오 설명

programmers

ROS 예제코드 분석

답안 설명

파이썬 예제코드 : teacher.py 소스코드

```
1   #!/usr/bin/env python
2
3   import rospy
4   from std_msgs.msg import String
5
6   rospy.init_node('teacher')
7
8   pub = rospy.Publisher('my_topic', String)
9
10  rate = rospy.Rate(2)
11
12  while not rospy.is_shutdown():
13      pub.publish('call me please')
14      rate.sleep()
```

파이썬 예제코드 : teacher.py

Xytron

- (1) #!/usr/bin/env python

  - 파일의 첫 줄에 #! 로 시작되는 라인을 Shebang 라인이라고 한다.
    - 스크립트 파일의 첫 줄에 사용된다.
    - 해당 파일의 실행에 어떤 인터프리터를 사용할지 지정한다.
    - PATH 환경변수에서 가장 우선되는 인터프리터를 찾아 해당 스크립트 파일을 실행한다.

  - 파이썬의 경우
    - 파이썬에서 # 로 시작되는 라인은 코멘트로 간주되므로 소스코드에 미치는 영향은 없다.
    - PATH 환경변수에서 가장 우선되는 python 바이너리를 찾아 해당 파이썬 파일을 실행한다.
    - 터미널에서 $ python teacher.py 명령으로 실행시킬 수도 있지만
    - Shebang 라인이 들어간 경우엔 그냥 $ ./teacher.py 형식으로 직접 실행시킬 수 있다.
    - 파이썬 버전을 구분해서 지정할 때도 사용
      - #!/usr/bin/env python2.6
      - #!/usr/bin/env python3

파이썬 예제코드 : teacher.py

Xytron

- (3) import rospy
  - rospy라는 라이브러리를 import해서 사용하겠다 라는 뜻이다.
  - import 키워드는 모듈, 패키지, 파이썬 표준 라이브러리 등을 가져온다.
    - 모듈은 특정기능을 python 파일 단위로 작성한 것이고,
    - 패키지는 특정 기능과 관련된 여러 모듈을 묶은 것이다.
    - 파이썬 표준 라이브러리는 파이썬에서 기본적으로 설치된 모듈과 패키지를 묶은 것이다.
  - rospy는 ROS의 파이썬 클라이언트 라이브러리이다.
    - rospy를 이용하면 파이썬 프로그래머들이 빠르게 ROS Topics, Services, Parameter의 interface를 사용할 수 있다.
    - 파이썬으로 ROS 프로그래밍을 할 때 필수적인 라이브러리이다.
    - rospy는 실행속도 보다는 구현의 편의성을 더 중요하게 생각하였기 때문에 빠르게 prototype을 만들 수 있다.

파이썬 예제코드 : teacher.py

Xytron

- (4) from std_msgs.msg import String

  - from import는 모듈의 일부를 가져올 수 있는 키워드이다.
    - ▶ from 뒤에 모듈 이름을 지정하고 import 뒤에 가져올 변수, 함수, 클래스를 입력한다.
    - ▶ import 뒤에 여러 개를 넣어도 된다.

  - 위의 경우 std_msgs.msg 라는 모듈에서 String 관련 부분만 가져와 사용하겠다는 의미이다.

  - import로 모듈을 불러올 때 사용자가 직접 별명(모듈 이름)을 설정할 수 있다.
    - ▶ 별명을 붙이려면 'import math as 별명' 처럼 import 뒤에 as를 붙여 이름을 지정하면 된다.
    - ▶ import numpy as np → numpy 라는 모듈을 사용할텐데 그 이름을 np로 해서 사용하겠다...

파이썬 예제코드 : teacher.py

Xytron

- (6) rospy.init_node('teacher')
  - 해당 노드를 초기화 하고 노드의 이름을 'teacher'로 한다는 코드이다.
  - 많은 노드들을 관리하고 통합하는 것이 ROS 프레임워크가 하는 일이고 그것을 python으로 만든 것이 rospy 라이브러리이다.
  - init_node는 rospy에서도 기본적인 함수이며 이 함수를 사용하여 생성된 노드는 다른 노드와 통신하면서 topic을 주고 받을 수 있다.
  - ROS 시스템 상에서 노드들이 topic을 주고받기 위해서는 노드에 고유의 이름을 할당해야 한다.
    - ▶ 위 코드의 경우에는 노드의 이름이 'teacher' 가 된다.
    - ▶ 노드의 이름에는 '/' slash와 같은 namespaces를 포함할 수 없다.

파이썬 예제코드 : teacher.py

Xytron

- (6) rospy.init_node('teacher')
  - init_node() 함수를 자세히 살펴보면
    - ▶ def init_node (name , argv = None , anonymous = False , log_level = None , disable_rostime = False , disable_rosout = False , disable_signals = False , xmlrpc_port = 0 , tcpros_port = 0):
  - 첫번째 인자는 Node의 이름이다. 타입은 string이다.
  - 두번째 인자는 argv인데 사용자가 지정한 argument를 넘겨받을 때 사용. 타입은 string의 list이다.
  - 세번째 인자는 anonymous라는 것인데 default가 False로 되어 있다. 만약 True 라면 노드의 이름이 자동으로 생성된다. (name 뒤에 임의의 숫자를 붙여준다)
    - ▶ 사용자가 같은 노드의 여러 instance를 원하고, 실제 이름에 신경 쓰고 싶지 않을 때 유용하다.
  - 네번째 인자는 log_level이다. 타입은 int 이며, default로 INFO 레벨이다.
    - ▶ rospy.DEBUG, rospy.INFO, rospy.ERROR, rospy.WARN, rospy.FATAL 등을 사용할 수 있다.

파이썬 예제코드 : teacher.py

Xytron

- (6) rospy.init_node('teacher')

  ▶ def init_node (name , argv = None , anonymous = False , log_level = None , disable_rostime = False , disable_rosout = False , disable_signals = False , xmlrpc_port = 0 , tcpros_port = 0):

  - 다섯번째 인자는 disable_rostime이다. 내부적인 테스트에만 사용된다.
  - 여섯번째 인자는 disable_rosout이다. 내부적인 테스트에만 사용된다.
  - 7번째 인자는 disable_signals이다.
    - ▶ True라면, rospy는 사용자의 signal handler를 등록하지 않는다.
    - ▶ 사용자가 main thread로부터 init_node를 call하지 않을 때나, 혹은 사용자가 자신만의 signal handling을 설정해야하는 환경에서 rospy를 사용할 때 이 flag를 셋팅해야 한다.
  - 8번째 인자는 xmlrpc_port이다. client XMLRPC node에 대한 포트번호이다.
  - 9번째 인자는 tcpros_port이다. TCPROS server는 이 포트를 통해 통신하게 된다.

파이썬 예제코드 : teacher.py

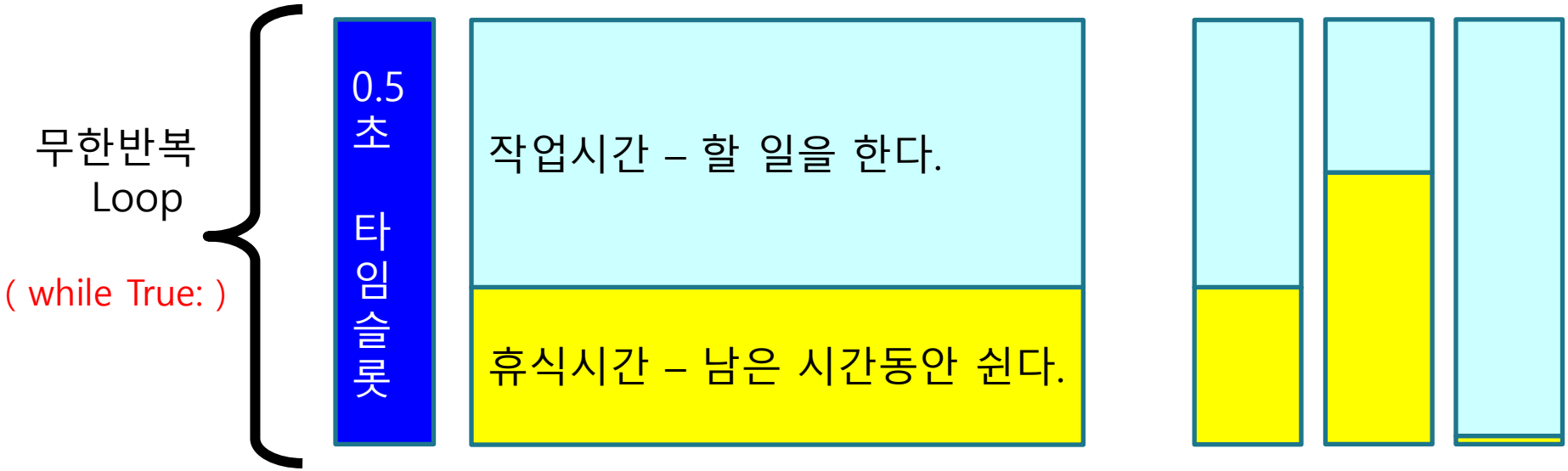
Xytron

- (8) pub = rospy.Publisher('my_topic', String)
  - 'my_topic' 이라는 이름의 토픽을 발행하겠다고 ROS 시스템에 자신을 publisher 로 등록하는 부분이다.
  - 그리고 'my_topic' 이라는 토픽에 담는 메시지의 데이터 타입은 String 이다 라고 지정하는 부분이다.
  - String은 위에서 선언했던 "from std_msgs.msg import String"에서의 String이다.
  - 나중에 String 타입의 메시지를 담아서 'my_topic' 이라는 이름의 토픽을 발행하려면 아래와 같이 코딩하면 된다.

    ▶ pub.publish('call me please')

파이썬 예제코드 : teacher.py

- (10) rate = rospy.Rate(2)
  - 1초에 2번 loop를 반복할 수 있도록 rate라는 객체를 만드는 코드이다.
    - ▶ 1초 안에 2번 반복 = 0.5초에 한번씩 루프를 돌아야 한다는 의미
    - ▶ 0.5초에 루프를 한번씩 돌기 위해서 0.5초짜리 타임슬롯을 만든다
    - ▶ loop 안에 있는 타임슬롯에 할당된 시간을 모두 소모한 후에 다시 loop를 반복



파이썬 예제코드 : teacher.py

Xytron

- (12) while not rospy.is_shutdown():
  - rospy.is_shutdown()이 True가 될 때까지 while loop안에 있는 실행문들을 반복하겠다는 뜻이다.
  - rospy.is_shutdown() 함수는 rospy 내부의 shutdown_flag를 검사한다.
  - 간단하게 ROS 시스템이 shutdown 되었는지의 여부를 검사하는 함수이다.

파이썬 예제코드 : teacher.py

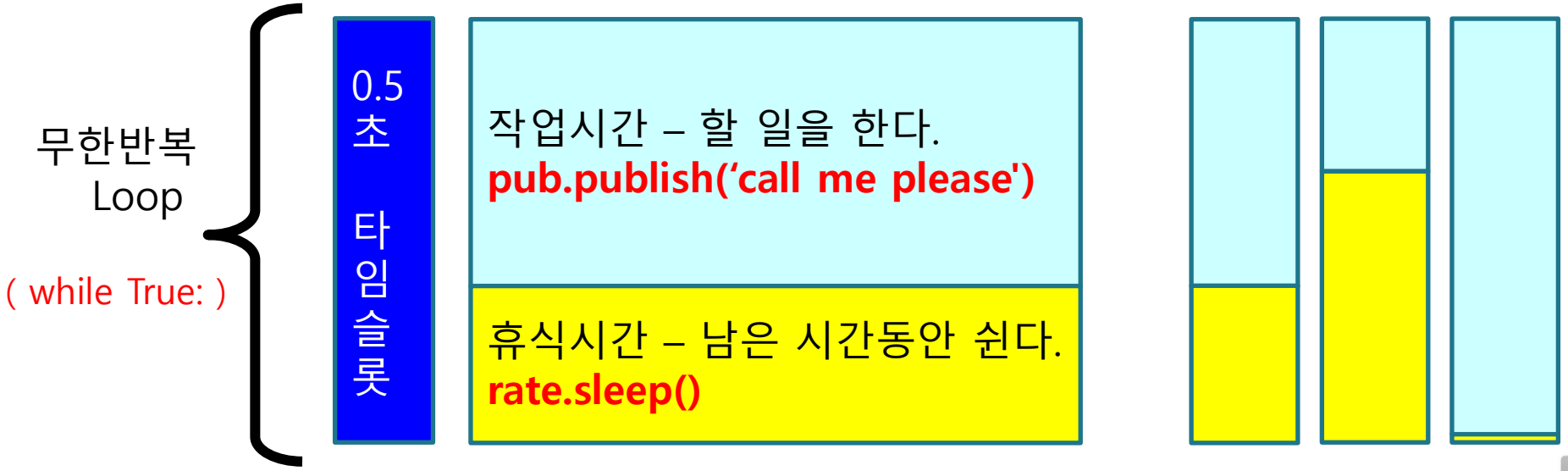
Xytron

- (13) pub.publish('call me please')
  - pub은 (8) 코드에서 'my_topic'라는 이름의 토픽을 발행하기 위해 만든 Publisher 인스턴스이다.
  - publish는 토픽에 데이터를 담아서 발행하는 기능을 수행한다.
  - publish는 2가지의 Exception이 일어나는데
    - ROSException는 rospy 노드가 initialization 되지 않았을 때 생기는 에러이다. 노드는 init_node 함수로 이름을 할당해 주어야 한다.
    - 또 다른 에러는 ROSSerializationException이다. 이것은 message를 serialize할 수 없을 때 일어난다. 이것은 보통 type error일 때가 많다.

파이썬 예제코드 : teacher.py

- (14) rate.sleep()

  - rate는 (10)번째 코드 rate=rospy.Rate(2) 에서 만들어진 인스턴스이다.
  - rate.sleep()는 while 루프 안에서 호출된다.
  - rate.sleep() 코드 때문에 while 루프가 0.5초마다 (1초에 2번) 반복된다.
  - 결과적으로 while 안에 있는 pub.publish 코드가 0.5초에 한번씩 동작하게 된다.



(정리) teacher.py 코드

```
1   #!/usr/bin/env python
2
3   import rospy
4   from std_msgs.msg import String
5
6   rospy.init_node('teacher')
7
8   pub = rospy.Publisher('my_topic', String)
9
10  rate = rospy.Rate(2)
11
12  while not rospy.is_shutdown():
13      pub.publish('call me please')
14      rate.sleep()
```

파이썬 코드이므로 python 앱으로 실행시켜야 한다...

rospy랑
String을 여기서 사용하겠다...

'teacher' 라는 이름을 갖는 노드를 하나 생성해라...

'my_topic' 라는 이름의 토픽을 발행할거다...
그 안에 들어갈 데이터는 String 타입이다...

아래서 루프 안의 코드가 1초에 2번 반복할 수 있도록...
타입슬롯을 0.5초 할당하겠다...

죽을 때까지 루프를 계속 돌아라...

'call me please' 값을 토픽에 담아 발행해라...

0.5초 지날때까지 잠깐 쉬어라(sleep) 해라...

파이썬 예제코드 : student.py 소스코드

```
1   #!/usr/bin/env python
2
3   import rospy
4   from std_msgs.msg import String
5
6   def callback(msg):
7       print msg.data
8
9   rospy.init_node('student')
10
11  sub = rospy.Subscriber('my_topic', String, callback)
12
13  rospy.spin()
14
```

파이썬 예제코드 : student.py

- (1) #!/usr/bin/env python
- (3) import rospy
- (4) from std_msgs.msg import String
  - teacher.py 에서와 동일한 코드이다 – 설명 생략합니다.

파이썬 예제코드 : student.py

Xytron

- (6) def callback(msg):
  - ‘callback’ 이라는 이름의 함수를 정의한다는 뜻이다.
  - python에서 함수를 정의하는 방법은 def 키워드를 사용하는 것이다.
    ▶ ‘def function_name(parameter):’ 로 함수를 선언하고 그 밑에 그 함수에 대한 코드를 작성한다.
  - 함수를 사용하고자 할 때는 우선 함수를 선언하고 나서 함수 호출을 원하는 곳에 ‘function_name(parameter)’ 코드를 넣으면 된다.
  - 함수를 정의할 때, 함수 호출보다 위에 있어야 한다. 만약 밑의 코드를 실행한다면 정의되지 않은 함수를 호출하였다고 에러가 난다.

파이썬 예제코드 : student.py

Xytron

- (7) print msg.data
  - 이 코드는 msg.data를 화면에 출력하는 코드이다.
  - print 함수는 python의 기본적인 함수로써 사용자가 어떤 값을 콘솔화면으로 출력할 수 있도록 해 준다.
  - rospy는 공식적으로 python2 버전을 사용한다. python2에서는 print문을 사용할 때 괄호를 사용하지 않는다.
  - 프린트 작업시 object들을 string으로 내부적으로 바꿔주므로 변수를 넣어도 string으로 변환되어 출력이 가능하다.

파이썬 예제코드 : student.py

- (9) rospy.init_node('student')
  - teacher.py에 똑같은 함수가 사용되므로 그 부분의 설명으로 대체한다.
  - 이 코드를 통해 'student'라는 이름의 노드가 새로 만들어진다.

파이썬 예제코드 : student.py

Xytron

- (11) sub = rospy.Subscriber('my_topic', String, callback)
  - 이번에 만든 노드는 토픽을 받는 구독자(Subscriber)임을 선언한다.
  - 받고자 하는 토픽의 이름은 'my_topic' 이다.
  - 그 토픽 안에 담긴 데이터는 String 타입이다.
  - 토픽이 도착할 때마다 'callback' 함수를 실행시킬 것을 ROS 시스템에 요청한다.
  - 결과적으로 토픽(메시지)을 받을 때마다 callback 함수가 한번씩 호출된다.
  - 해당 callback 함수는 앞서의 (6)처럼 미리 정의되어 있어야 한다.

파이썬 예제코드 : student.py

Xytron

- (13) rospy.spin()
  - rospy.spin()은 ROS 노드가 shutdown될 때까지 Block하는 함수이다.
  - 즉, shutdown signal을 받을 때까지 무한루프에 들어가게 된다.
  - 이 무한루프에서는 topic을 받거나 time triggering 같은 이벤트가 발생하면 callback 함수가 호출되지만 그렇지 않으면, sleep 상태가 된다.
  - 사용자의 노드가 callback 이외에 어떤 일도 하지 않는다면 spin() 함수를 사용해야 한다.
  - rospy.spin()과 비슷한 일을 하는 rospy.sleep()도 있다.
    - ▶ rospy.sleep()과 rospy.spin()의 차이점은 sleep()은 특정한 시간이 주어지고 그 시간 동안만 sleep 상태가 유지된다.

(정리) student.py 코드

```
1   #!/usr/bin/env python
2
3   import rospy
4   from std_msgs.msg import String
5
6   def callback(msg):
7       print msg.data
8
9   rospy.init_node('student')
10
11  sub = rospy.Subscriber('my_topic', String, callback)
12
13  rospy.spin()
14
```

파이썬 코드이므로 python 앱으로 실행시켜야 한다...

rospy랑
String을 여기서 사용하겠다...

함수 선언 ...
msg 안에 담긴 데이터를 꺼내 화면에 출력해라...

'student' 라는 이름을 갖는 노드를 하나 생성해라...

'my_topic' 라는 토픽을 받을 거다...
토픽 안에 담긴 데이터는 String 타입이다...

토픽이 도착하면 'callback' 이름의 함수를 호출해라...

죽을 때까지 여기서 혼자 루프를 계속 돌아라...

(정리) student.py와 teacher.py의 상호 동작

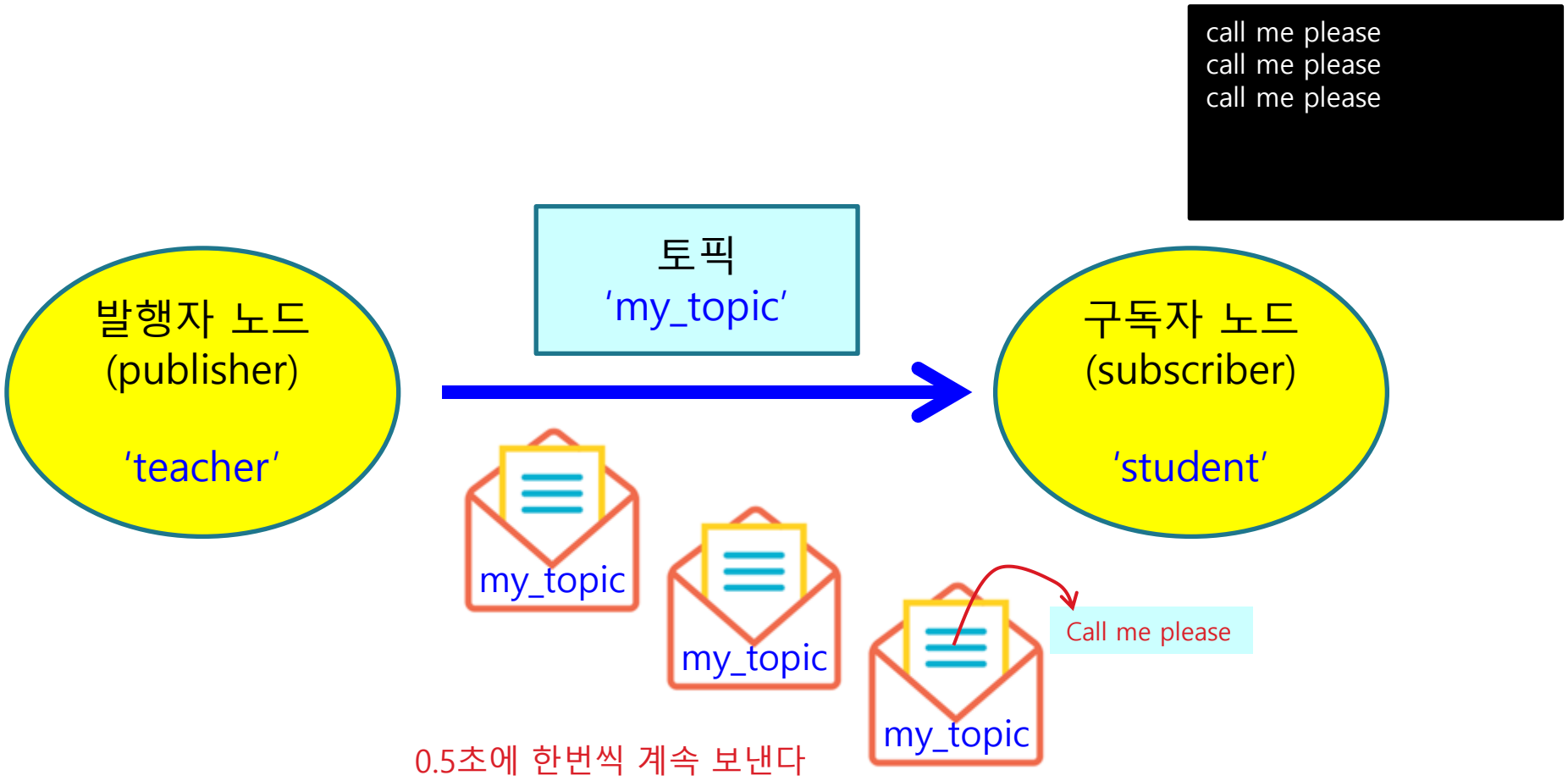
Xytron

- teacher.py에서는
  - node의 이름을 'teacher'로 설정한다.
  - 'my_topic'라는 이름의 topic을 발행하는 걸로 설정한다.
  - 'call me please'라는 message를 발행한다.
  - 그리고 1초에 2번 반복하게끔 sleep 하는 함수를 사용한다.
  - 이걸 node가 종료될 때까지 무한반복한다. (결국 계속해서 1초당 2번씩 토픽(메시지)를 보낸다)

- student.py에서는
  - node의 이름을 'student'로 설정한다.
  - 'my_topic'라는 Topic을 구독하는 걸로 설정한다.
  - callback 함수를 정의하여 'my_topic' 토픽이 올때마다 그 안에 담긴 data를 출력하게끔 준비한다.
  - 무한반복 루프에 들어간다.

- 동작 결과
  - Teacher 노드가 'my_topic'라는 이름의 topic에 "call me please" 문자열을 담아 보내면, student 노드가 해당 topic을 받아 그 안에 담긴 문자열을 꺼내 화면에 출력한다.

(정리) student.py와 teacher.py의 상호 동작



Q&A

감사합니다.

㈜자이트론
허성민
smher@xytron.co.kr