|  |  |
| --- | --- |
| 년도-학기 | 2021년 2학기 |
| 과목명 | 자동화프로그래밍 |

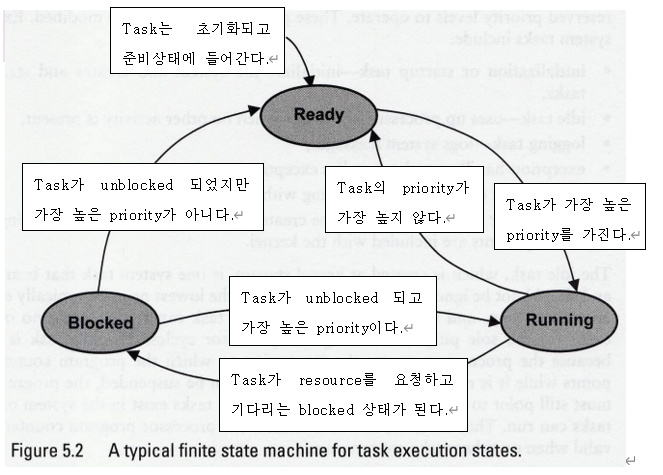
|  |  |
| --- | --- |
| **번호** | **실험 제목** |
| LAB 9 | PD Control with FreeRTOS |

|  |  |
| --- | --- |
| 실험 일자 | 2021년 11월 16일 |
| 제출자 이름 | 강\*\*\*\* |
| 제출자 학번 | 201803\*\*\*\* |
| 팀원 이름 |  |
|  |  |

**Chapter 1. 관련 이론(Theoretical Background)**

RTOS는 Real Time Operating Systems의 약자로 실시간 운영 체제를 의미하고 주어진 작업을 정해진 시간 안에 수행할 수 있는 환경을 제공한다.

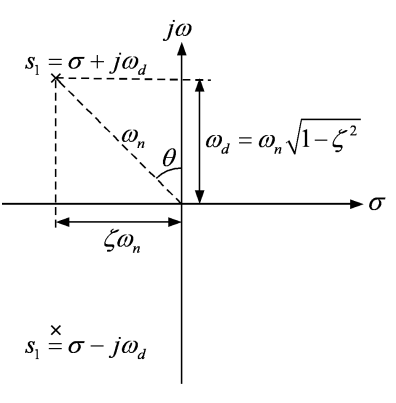
RTOS의 task는 실행의 가장 기본적인 단위를 말한다. ready, running, blocked의 3가지 state로 구분되어 다음과 같은 동작을 보인다.



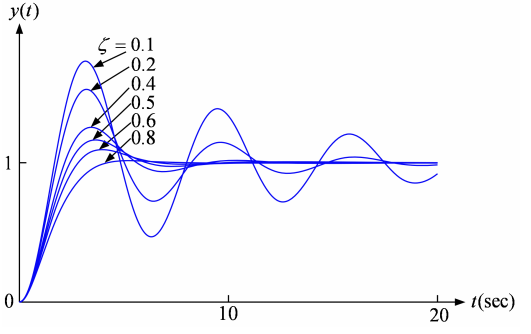
Scheduler는 ready 상태에 있는 여러 task 중 어떤 task를 실행할지 결정한다. Scheduling Algorithms에는 round-robin 방식과 preemptive priority 방식이 있다. Round-robin 방식은 일정 시간 단위동안 돌아가면서 task를 실행한다. Preemptive priority 방식은 높은 priority를 가진 task가 우선적으로 실행되도록 한다.

2차 시스템 전달함수의 표준형은 다음과 같다. 이 때 분모를 0으로 만드는 s 값은 pole이 되며 pole의 위치에 따라 시스템의 응답 특성이 달라진다.

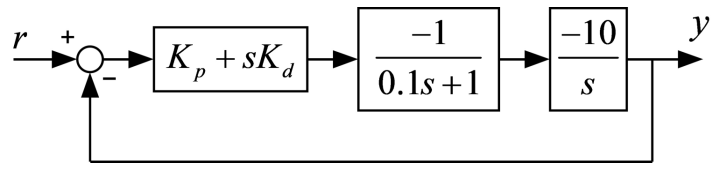
테이블이(가) 표시된 사진

자동 생성된 설명 

ζ는 감쇠비, damping ratio로 그 크기가 클수록 overshoot이 작아진다.



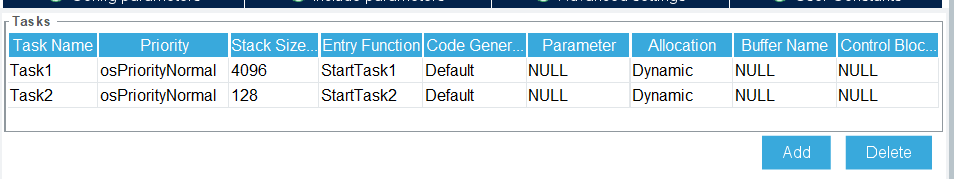
- PD 제어기



비례 제어만을 사용할 때의 overshoot을 줄이고 급격한 변동을 제어한다. 이전 결과와의 편차를 파악해 출력 값을 조절하여 제어한다. 이에 따라 안정성이 향상된다.

**Chapter 2. 실험 결과(Experimental Results)**

먼저 같은 priority를 갖는 Task1과 Task2를 설정한다.



기존 프로그램의 전역변수를 사용하면서 Task1은 시리얼 터미널에 실습 데이터를 프린트하는 기능을 실행할 수 있도록 아래와 같이 코드를 작성하였다.

텍스트이(가) 표시된 사진

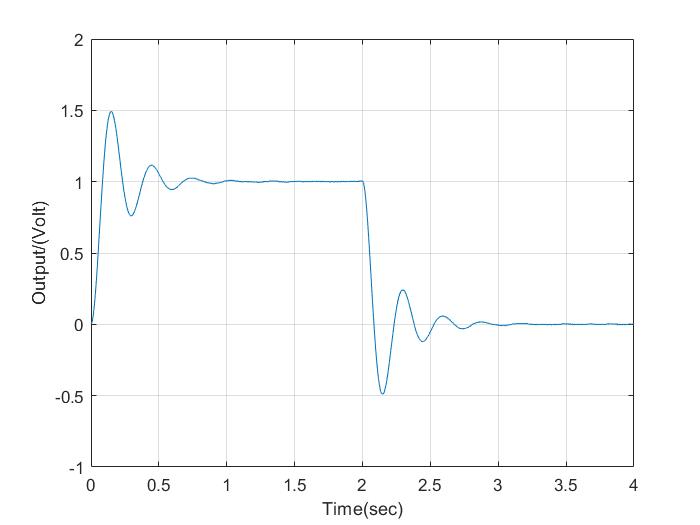
자동 생성된 설명

Task2는 버튼 입력을 polling방식으로 감지하는 기능을 한다. 버튼이 눌리면 1을 반환하고, 눌리지 않으면 0을 반환하는 함수를 통해 그 값이 0에서 1로 변화할 때 버튼이 눌린 것임을 조건으로 하여 data\_flag=1이 되도록 하였다.

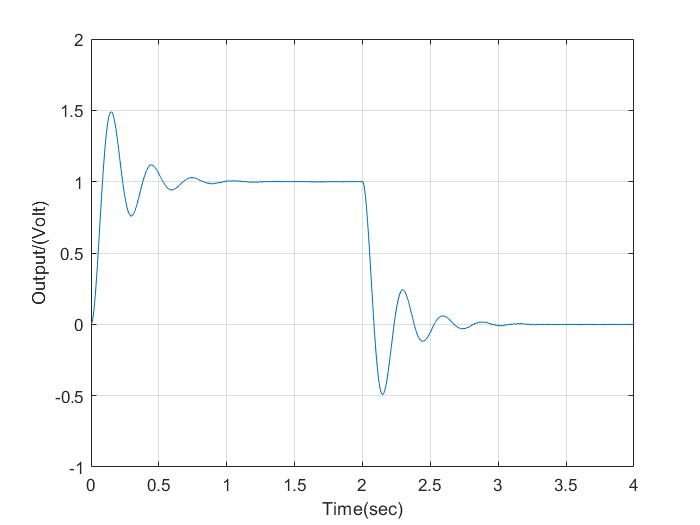
텍스트이(가) 표시된 사진

자동 생성된 설명

원래의 Lab3 PD Control을 실행했을 때의 실험 결과를 plot하면 다음과 같다.



FreeRTOS를 사용하여 Task1과 2를 이용한 이번 실습의 실행결과를 plot한 결과는 다음과 같다. 이전 실습과 아주 비슷한 결과를 보여준다.



**Chapter 3. 결론 및 Discussion**

버튼 인터럽트 대신 task 2개를 사용하여 과제를 수행했다. 각 task가 수행할 코드를 작성할 때도 원래 프로그램에 구현된 코드가 있었기 때문에 과제에서 어려웠던 점은 없었다. Lab3 원래 코드의 실험 결과와 freertos를 사용한 실험 결과를 비교해보면 거의 같은 사진으로 보일 정도로 비슷한 결과가 나왔다.

하지만 semaphore나 mutex를 이용해서 과제를 해결하는 것은 어려웠고 성공하지 못했다. 기존 코드는 data\_flag에 0,1,2 data\_done에 0,1을 대입하여 상황을 구분하는 것에서 semaphore나 mutex로 총 다섯가지의 상황을 어떻게 구별할 수 있을 지 생각해보았다. 다섯개의 상황에 순서가 있는 것은 분명하고, 기존 코드를 살펴보며 상황을 정확히 이해하였지만 적재적소에 semaphore나 mutex를 사용하는 것에 실패하여 결국엔 해결하지 못해 아쉬움이 남았다.

**Appendix: main 소스 코드 별도 파일로 첨부**