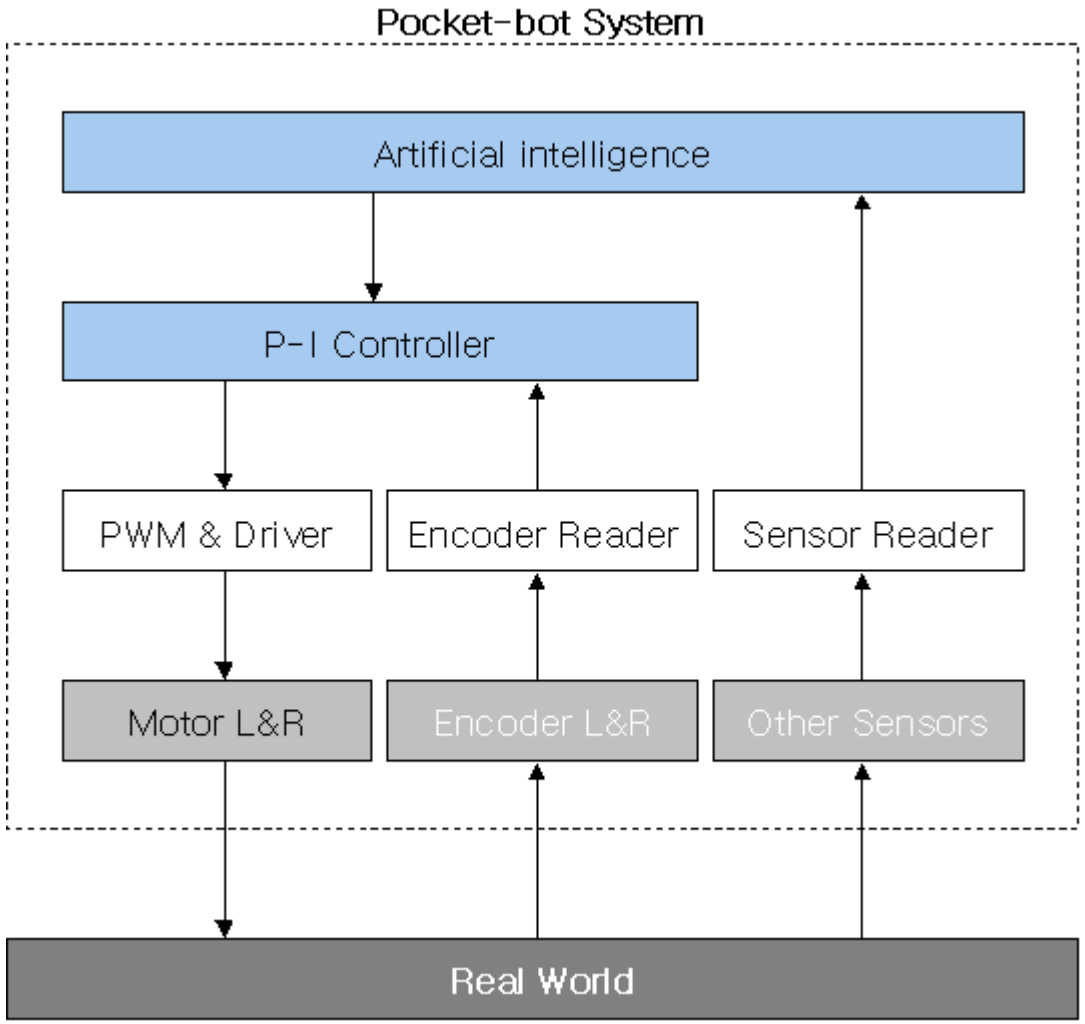
# < State Machine >

뭐 학술적인(?) 이야기는 집어치울께요. 저도 잘 모르니까요. 제목을 너무 거창하게 붙인게 아닌지 걱정도 되네요.

일단 포켓봇 시스템을 하나씩 뜯어서, 레이어(Layers)별로 늘어놓아 봅니다.




네... 바깥 세상과 포켓봇과의 인터페이스는 역시 센서랑 엔코더, 그리고 모터로군요. 흰색 글씨는 입력받는 놈이라서 그렇게 했습니당...

그 바로 윗 레이어인 흰색 블록들은... 이를테면, 우리몸의 자율신경계 쯤에 해당되겠죠. 어떤 경우에 있어서든지 항상 작동하고 있어야 한다고 보면 될테니까요.

그리구, 또 그 바로 윗 레이어인 하늘색 블록 - PI Controller 는 이를테면 '하위제어기'라 할 수 있겠습니다. '상위제어기'인 인공지능으로부터 지령을 받아 단순한 작업만 도맡아 하고 있으니까요. 여기서는 오로지 속도제어만 열라게 하죠? 이건 근육을 통제하는 쪽 신경계 쯤 되겠군요...

가장 상위의 '인공지능'이라고 거창하게 이름붙인 부분은, 받아들인 데이터들을 토대로 로봇의 행동을 결정하고 지령을 내려주는 부분이라 할 수 있겠죠. MPU의 성능이 충분하고 메모리가 무한정이라면... 설계자의 능력에 따라 얼마든지 복잡한 인공지능을 심어넣을 수 있겠다는 생각도 드네요... 물론 우리몸의 두뇌에 해당되겠군요.

근데 이 시스템을 훑어 보면, 한가지 특성을 엿볼 수 있겠습니다. 즉 무한루프라는 것이죠.

Encoder Reader 는 전의 글에서 고찰해본 바와 같이, 가장 짧은 시간단위로(220 Hz 이상) 엔코더의 값을 읽어들이구요...

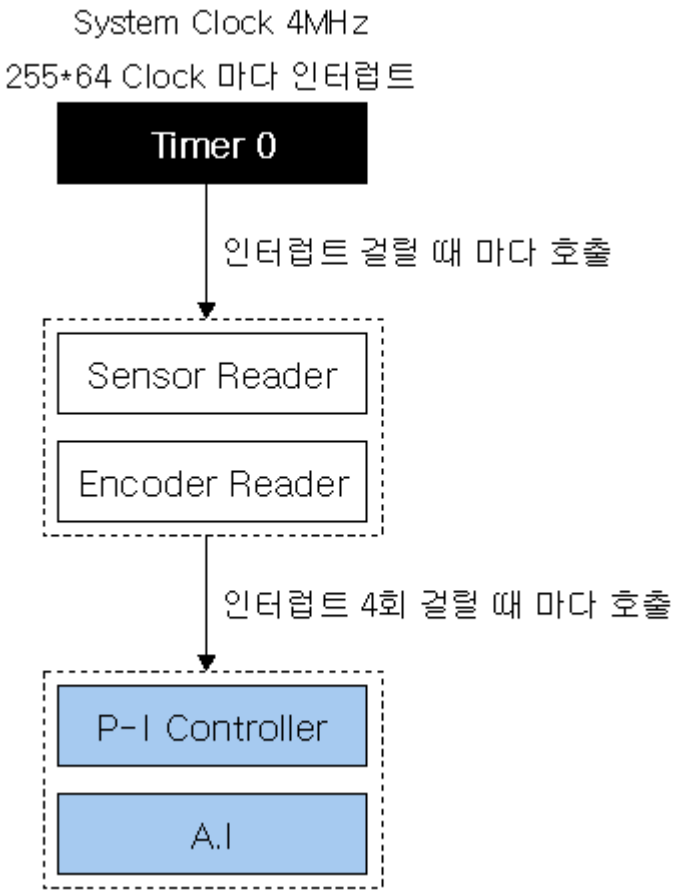
P-I Controller 는 이보다 좀 더 느린 시간단위로(50 Hz 이상정도?) 반복적으로 출력값을 결정, 뱉아내어 주구요...

상위제어기는 또 이와 같거나 더욱 느린 시간단위로(20 Hz 이상 정도? 또는 Event Driven으로) 역시 처리하게 하면 되겠군요.

암튼 모두 무한루프라고 할 수 있겠습니다. 게다가 이것들은 '일정시간 단위로' 주기적으로 실행되어야 한다는 실시간성(Real Time Feature)을 확보해야 겠습니다.

이게 뭐냐구요? 이게 바로 상태기계(State Machine)가 아닐까요?

8515에서 실시간성을 확보하기 위해서는 당근 타이머를 사용해야겠죠. 현재 엔코더 읽기로 0번을, PWM 발생용으로 1번을 모두 사용하고 있는 상태입니다. 그러면 제어기를 위한 타이머는 따로 남은게 없으니 0번 타이머가 불러내 주는 인터럽트를 더 활용하는 방법을 써야 겠습니당.




요런 식으로 말이죠. '255*64 Clock' 이란 말은, 타이머 기본 클럭을 64분주 해주고 여기다가 8비트 카운터 오버플로우 인터럽트를 걸게 되니 나오는 값이죠... 게다가 제어기로 넘어가면 약 4회 마다 한 번 실행되는 꼴이니... 총 클럭은 255*64*4 = 65280 Clock 이 되는 군요. 뭐 상당히 난잡한 코드로 된 복잡한 제어기라 하더라도, 6만 클럭 안에 실행을 완결 짓는건 가능하겠죠... 쩝. 대략 가늠해 본 것입니다.

ps. 음. 시스템의 윤곽이 드러났으니... 해야될 일의 순서도 명확해 집니다.

다음엔 이러한 밑그림에 바탕을 두고 P-I 제어기를 구성해서 테스트해 보구요, 그담엔 IR 센서를 달고... 마지막으로 상위제어기를 달아 붙이면 그대로 포켓봇의 라인트레이싱 기능은 완성이로군요.

이글 읽으시는 분들, 새해 복 많이 받으시길 바랍니다!