<마이크로로봇 과제 전류지령 생성 PC 프로그램 파일 설명>

2016.03.29

서울대학교 전기에너지변환연구실

작성자 : 이준

<rt\_setting.h>

프로그램의 주기적 동작 및 rtai 관련 설정들을 담은 파일.

<variables.h>

프로그램에서 사용되는 상수와 변수들의 선언. 프로그램의 동작 방식에 대한 설정을 #define으로 선언된 부분을 수정하여 할 수 있다. (연산 중 결과 출력 여부, UDP 통신 receive 시 timeout 시간 설정 등)

<variables.cpp>

변수들의 선언. variables.h에서 마저 설정되지 않은 동작 방식들을 지정한다. 주로 전류 지령 계산법, 위치와 각도 estimator 동작 방식을 설정한다. UDP 통신에 관련된 정보(ip, port 등)도 이 파일에서 설정한다.

<microrobot\_pc\_app.cpp>

실행파일이 만들어 지는 메인 cpp 파일. rtai, UDP 통신 설정 등 초기화를 수행한 뒤 중앙의 while문 안을 1ms 주기로 반복하도록 되어있다. 프로그램을 마칠 때 실행 결과에 대한 간단한 통계를 출력한다.

[int rt\_setting]

1ms 주기로 UDP 통신과 전류 지령 계산이 반복되도록 rtai의 설정을 한다. 실패했을 경우 -1을 반환.

[int timer\_precision\_test]

rtai의 timer 가끔 비정상적으로 설정되는데 이것이 잘 설정되었는지 확인하는 test 함수. 비정상일 경우 -1을 반환.

[int sock\_setting]

UDP 통신을 variables.cpp에서 설정한 ip, port 정보로 준비. 잘못 설정된 경우 -1을 반환.

[void init\_vectors]

위치, 정렬, 속도, 지령, 센싱 정보 등의 행렬을 초기화.

[void reset\_current\_ref]

전류 지령과 센싱 정보 행렬을 0으로 reset.

[void periodic\_recv]

매 주기 실행되는 UDP 수신 함수. variable.h 에서 설정된 timeout 시간을 넘도록 수신한 내용이 없으면 timeout\_cnt를 늘리며 이것이 1000회(1초)가 되면 통신이 끈겼다고 판단한다. 한 번 수신을 성공한 경우 혹시 버퍼에 더 남아있는 정보가 있는지 확인하여 버퍼를 비울 때 까지 읽어 들여 최근 정보를 사용한다. recv\_buf 라는 문자열 버퍼에 자료를 수신한 뒤 이것을 현재 동작 모드에 따라 적절히 해석(udp\_func.cpp 참고)한다.

[void periodic\_send]

매 주기 실행되는 UDP 송신 함수. 현재 PC의 모드와 상태에 따라 UDP 자료를 만들어 송신한다. UDP 자료를 만들며 1초마다 현재의 모드와 상태를 출력한다. 전류 지령을 계산하는 모드에 있는 경우 get\_current\_ref 함수를 실행하고 이것이 100us 이상 걸렸을 경우 보고한다.

[void mode\_status\_check]

수신한 데이터를 분석하여 하위 보드들의 모드와 상태 설정의 변화를 해석하여 PC의 모드와 상태를 수정한다.

[void clear\_recv]

recv\_buf에 있는 내용을 비워 안정적인 수신이 가능하도록 하는 함수.

[void fprint\_data]

프로그램의 결과를 csv 파일로 출력한다. 전류 제어를 하는 순간에만 기록하며 1ms 간격으로 기록한다. 자기장(토크)와 힘 지령, 전류 지령, 추정된 위치와 각도(자화), 측정된 전류, recv timeout 여부를 기록한다.

<udp\_func.cpp>

UDP에 사용된 datagram을 통해 데이터를 만들거나 해석하기 위한 함수를 모아놓은 파일.

[void AddSendHeaderData]

PC의 현재 모드와 상태 그리고 통신 정상 여부 확인용 신호를 담아 header를 만든다.

[void init\_mode\_udp\_send\_func, void inv\_test\_mode\_udp\_send\_func]

INIT 모드와 INV\_TEST 모드에서는 PC가 header 이외에 보내는 데이터가 없음.

[void idle\_mode\_udp\_send\_func]

IDLE 모드에서 PC가 보내는 데이터로 전류 지령을 0으로 보낼 것인지 (RUN 모드에서의) 마지막 전류를 유지할 것인지를 알려준다.

[void run\_mode\_udp\_send\_func]

RUN 모드에서 PC가 보내는 데이터로 전류 지령을 담아 보낸다.

[void init\_mode\_udp\_recv\_func]

INIT 모드에서 PC가 받은 데이터를 해석. 받은 정보를 통해 준비가 안되거나 통신이 불안정한 인버터 보드가 있는 경우 출력한다. 출력된 정보는 16진수 두 자리로 출력이 되는데 이를 2진수로 풀어(LSB가 1번, MSB가 8번 인버터보드) 몇 번째 보드가 문제인지 확인할 수 있다.

[void inv\_test\_mode\_udp\_recv\_func]

INV\_TEST 모드에서 PC가 받은 데이터를 해석. 전류나 전압 fault가 발생한 인버터를 위와 같은 방법으로 출력.

[void idle\_mode\_udp\_recv\_func, run\_mode\_udp\_recv\_func]

IDLE, RUN 모드에서 PC가 받은 데이터를 해석. 전류 전압 fault 여부와 측정 전류를 수신.

[void read\_udp\_header]

통신용 보드가 보내는 header 정보를 해석. 하위 시스템의 모드와 상태를 알 수 있고 이에 따라 PC의 모드와 상태를 적절히 변경.

[void print\_fault\_INVs]

전류를 흘리는 모드에서 전압이나 전류에 fault가 있는 인버터보드를 판별하여 출력하는 함수. 위의 수신 함수들에서 공통으로 사용.

[void read\_current\_sen]

IDLE, RUN 모드에서 수신한 전류 정보를 해석하여 행렬에 저장하는 함수.

[void copy\_current\_ref\_to\_sen]

UDP 수신이 정상적으로 이루어지지 않은 경우 old\_ref\_delay 만큼 이전의 전류 지령을 현재 수신한 측정 전류에 대입하는 함수. 수신을 못하였어도 위치와 각도를 계속 추정해나가기 위하여 필요하다.

<data\_form\_change.cpp>

문자열로 받아지는 UDP 데이터를 float로 바꾸는 등 데이터 형변환을 수행하는 함수들일 필요하다. 혹은 반대로의 과정도 필요.

[void AddSendFloatdata(float f)]

송신을 할 전류 지령(float, 4바이트)를 4개의 문자(1바이트\*4)로 나누어 send\_buf에 담는 함수.

[int ReadRecvIntData]

현재 recv\_iter자리로부터 4개의 문자(1바이트\*4)를 읽어 int(4바이트)로 변환해 반환한다.

[float ReadRecvFloatData]

현재 recv\_iter자리로부터 4개의 문자(1바이트\*4)를 읽어 float(4바이트)로 변환해 반환한다. 보통 측정 전류를 읽을 때 쓰인다.

<load\_b\_lut.cpp>

data 폴더에 저장되어 있는 여러 자기장 look-up table들을 읽어와 행렬에 저장해 두는 함수. 현재 제어는 easy\_B\_LUT 파일을 통해 로봇이 중앙에 있으며 자기장에 정렬되어 있다고 가정하고 전류 지령을 생성한다. 위치와 방향을 정확히 아는 경우 B\_LUT 변수들을 사용해 정확히 연산할 수도 있다.

<actuation\_matrix\_calc.cpp>

Actuation matrix와 관련된 연산 함수들을 담고 있다.

[void bound\_floor(int \*floor\_val)]

현재의 좌표를 grid에 적절히 넣어 B\_LUT를 통해 자기장과 기울기를 연산할 수 있도록 준비해주는 함수.

[void calc\_b(vector<vector<double> > \*B\_LUT, Matrix \*Act, Matrix \*Act\_temp, int nth\_col, int x\_floor, int y\_floor, int z\_floor, double x\_ratio, double y\_ratio, double z\_ratio]

현재의 좌표와 한 코일의 look-up table을 받아 그 지점에 그 코일이 만들어내는 자기장과 기울기를 계산하여 주어진 actuation matrix의 원소를 채운다. 프로그램 설정에 따라 easy\_B\_LUT를 사용하기도 한다.

[void get\_actuation\_mtx(Matrix \*Act, bool get\_act\_mat\_ref\_mode)]

현재의 위치와 각도정보를 사용하여 actuation matrix를 만들어 내는 함수. 프로그램의 설정에 따라 자기장-힘 actuation matrix를 만들지 혹은 토크-힘 actuation matrix를 만들지가 결정된다.

[void get\_current\_ref]

주어진 자기장(토크), 힘 지령을 수행할 수 있는 전류 8개를 위에서 만든 actuation matrix의 pseudo inverse를 통해 계산하는 함수. 지령에 사용자의 manual force 지령을 더한 뒤 전류를 계산한다. 시스템의 안정적인 구동을 위하여 RUN 모드에 들어간뒤 처음 SOFT\_REF\_CNT 회 동안 선형적으로 지령을 증가 시킨다. (그 이후의 step 지령에는 무효.) timeout 시 사용하기 위하여 생성된 지령들을 Current\_ref\_old에 미뤄가며 저장한다.

[void update\_PM]

측정된 전류(혹은 old reference)를 통해 힘과 토크를 계산하고 이를 사용하여 위치와 각도를 갱신하는 estimating 함수. 전달함수에 따른 다른 estimating 방법들을 기재해 놓았음. 프로그램의 설정에 따라 자기장으로 각도를 갱신할 지 토크로 각도를 갱신할 지 선택이 된다.

<gen\_BF\_ref.cpp>

자기장과 힘 지령을 만들어내는 함수. 현재는 사용자가 선택한 종류의 지령을 open loop으로 출력하도록 되어있다. 자기장을 표현하는 변수들은 ‘B’를 포함하고 있으나 variable.cpp 에서 설정해주는 것에 따라 토크 변수로 사용할 수 있다.

이후 모션 제어 함수의 내용으로 대체되면 된다.

<user\_input\_thread.cpp>

사용자의 입력을 받는 thread. 힘 자기장 지령 변경, 지령 회전 주파수, manual force, 제어 인버터 개수 등 여러 변수의 변경을 프로그램 실행 중 할 수 있다.

<matrix.cpp>

Matrix의 곱셈 및 pseudo inverse 계산에 쓰이는 라이브러리. 수정하지 말 것.