**Microrobot 제어 코드 설치 및 실행 방법 매뉴얼**

작성 기관 : 서울대학교

작성일 : 2016년 8월 29일

작성자 : 이준

**<터미널 & 리눅스 명령>**

본 매뉴얼은 리눅스 명령들을 실행하는데 이는 리눅스 터미널에서 실행한다. Ubuntu desktop 에서 터미널을 여는 단축키는 ctrl+alt+T 이다.

리눅스 명령은 박스(표)로 테두리를 표시하였으며 각 명령 입력 후 엔터를 쳐서 실행하면 된다.

**<프로그램 폴더 만들기>**

제공된 압축 파일인 microrobot\_pc\_app.tar.gz 을 원하는 폴더에 놓고 해당 폴더로 이동한다.

권장 진행은 리눅스 계정 폴더에 ‘Project’ 라는 폴더를 만들어 그 안에서 작업을 시작하는 것이다.

|  |
| --- |
| cd /home/[리눅스계정이름]  mkdir Project  cd Project |

프로젝트 폴더에서 압축을 푼다.

|  |
| --- |
| tar zxf microrobot\_pc\_app.tar.gz |

**<프로그램 구성 파일>**

폴더 안에는 다음의 파일들이 있다. 프로그램의 동작 방식이나 주요 변수를 변경하는데 수정을 해도 큰 문제가 없는 파일은 파란색, 주의가 필요한 파일은 빨간색으로 표기하였다. 검정색으로 쓰여진 파일은 수정을 할 경우 프로그램의 실행에 큰 지장이 있는 파일들이므로 가능한 한 수정하지 않도록 한다.

|  |  |
| --- | --- |
| 파일명 | 설명 |
| actuation\_matrix\_calc.cpp | 힘과 자기장 지령을 전류 지령으로 변환하는 행렬인 actuation matrix 를 마이크로로봇의 상태(위치와 각도)에 따라 새롭게 계산하는 파일. |
| actuation\_matrix\_calc.h | 위 파일의 함수들을 정의하는 파일. |
| coil\_configuration\_4.csv | 4코일 시스템의 각 코일들을 정의하는 파일. 코일들을 평면상에서 등가 magnetic dipole 들로 모델링하여 기입한다. Magnetic dipole 4개의 위치와 착자 벡터를 순서대로 기입한다. (파일 내부에 기입하는 순서가 쓰여있다.) x, y 순서쌍을 tab으로 구분하지 않고 콤마(,)로 구분하여 기입되도록 주의한다. |
| coil\_configuration\_8.csv | 8코일 시스템의 각 코일들을 정의하는 파일. 코일들을 공간상에서 등가 magnetic dipole 들로 모델링하여 기입한다. 기입 방법과 주의사항은 위와 같음. |
| current\_result.csv | 프로그램의 실행 결과가 기록되는 파일. 자기장 지령, 힘 지령, 전류 지령, 마이크로로봇 위치, 마이크로로봇 정렬 방향, 측정 전류, 통신 상태를 1ms마다 기록한다. (파일이 없는 경우 생성) |
| data\_form\_change.cpp | UDP 통신에 필요한 변수형 변환 함수들. |
| data\_form\_change.h | 위 파일의 함수들을 정의하는 파일. |
| easy\_B\_LUT\_4.csv | 4코일 시스템에서 actuation matrix 를 마이크로로봇의 위치 변화에 상관 없이 만들어 계산하고자 할 때 사용되는 고정 LUT. 각 코일에 1A의 전류를 흘렸을 때에 제어 영역 중심에 생성하는 자기장, x방향 편미분, y방향 편미분 값을 순서대로 기입한다. |
| easy\_B\_LUT\_8.csv | 8코일 시스템에서 actuation matrix 를 마이크로로봇의 위치 변화에 상관 없이 만들어 계산하고자 할 때 사용되는 고정 LUT. 각 코일에 1A의 전류를 흘렸을 때에 제어 영역 중심에 생성하는 자기장, x방향 편미분, y방향 편미분, z방향 편미분 값을 순서대로 기입한다. |
| gen\_BF\_ref.cpp | 자기장과 힘 지령을 1ms 간격으로 업데이트 하는 파일. 각 지령은 사용자가 ‘user\_input\_thread.cpp’ 에서 설정한 값을 바탕으로 만들어진다. 다른 종류의 지령을 만드는 경우, 지령을 다른 pc로부터 받거나 화상으로 입력 받는 경우 등에 파일을 수정한다. |
| gen\_BF\_ref.h | 위 파일의 함수들을 정의하는 파일. |
| limit\_methods.cpp | 생성된 전류 지령이 안전치를 넘는 경우 지령을 수정하는 방법들을 담은 파일. 다른 전류 지령 보정법을 적용할 경우 이 파일을 수정하지 않고 variables.cpp에서 관련 설정 변수를 수정한다. |
| limit\_methods.h | 위 파일의 함수들을 정의하는 파일. |
| load\_b\_lut.cpp | 같은 폴더에 있는 ‘coil\_configuration’ 파일들과 ‘easy\_B\_LUT’ 파일들을 불러와 프로그램 실행 중 사용하도록 하는 파일. |
| load\_b\_lut.h | 위 파일의 함수들을 정의하는 파일. |
| Makefile | 프로젝트를 컴파일하고 실행 파일을 만들기 위한 설정 파일. 새로운 프로젝트 파일이 생기거나 없어졌을 때에 수정한다. |
| matrix.cpp | 프로그램 실행 중 필요한 변수형과 연산들을 정의하는 파일. |
| matrix.h | 위 파일의 함수들을 정의하는 파일. |
| microrobot\_pc\_app | 실행 파일. 실행 방법은 다음 장에서 설명. |
| microrobot\_pc\_app.cpp | 메인 프로그램. 프로그램의 주요 실행 과정을 바꾸거나 테스트를 진행할 때에 수정한다. |
| microrobot\_pc\_app.o | RTAI 모듈과 연동 중 발생하는 출력 파일. |
| rtai-insmod.sh | RATI 모듈들을 background에서 실행하도록 하는 명령 집합. |
| rtai-rmmod.sh | RATI 모듈들을 실행 중지하도록 하는 명령 집합. |
| udp\_func.cpp | 하위 보드와 정보를 주고 받고 해석하는 파일. |
| udp\_func.h | 위 파일의 함수들을 정의하는 파일. |
| user\_input\_thread.cpp | 사용자의 입력을 받는 thread를 만드는 파일. 사용자 입력 키를 변경하거나 다양화 시키고자 할 때 수정한다. |
| user\_input\_thread.h | 위 파일의 함수들을 정의하는 파일. |
| variables.cpp | 실행에 관련된 변수들을 선언하고 값을 정하는 파일. 실행 중 동작을 변경하고자 할 때 이 파일을 수정한다. 제어법 수정에 관련된 변수들을 파일 상단에 모아두었으며 이외의 변수들은 수정 시 실행에 문제가 생길 수 있으므로 주의한다. |
| variables.h | 위 파일의 함수들을 정의하는 파일. 변수가 추가되거나 삭제된 경우 수정한다. |

**<프로그램 동작 설정>**

프로그램의 위치, 각도 업데이트 방법, 사용 코일 개수, 초기 지령 등의 설정은 variables.cpp 에서 한다. 파일 상단에 있는 주석 사이의 다음 변수들이 동작에 관련된 주요 변수들이며 다른 변수들은 수정 시 실행에 문제가 생길 수 있으므로 주의한다. 회색 배경으로 표시된 변수들은 variables.h 파일에 있고 나머지 변수들은 variables.cpp 파일에 있다.

|  |  |
| --- | --- |
| 변수 | 설명 |
| MASS\_DENSITY | 마이크로로봇의 질량 밀도. |
| FLUID\_DENSITY | 동작 환경(실리콘오일)의 질량 밀도. (부력 계산에 사용.) |
| MAGNETIZATION\_DENSITY | 마이크로로봇의 부피 자화 밀도 |
| eta | 동작 환경(실리콘오일)의 점성 계수 |
| x\_robot, y\_robot, z\_robot | 마이크로로봇의 직경(x, y 동일), 높이 변수 |
| drag\_translation, drag\_rotation | 병진, 회전 운동의 마찰 계수 |
| F\_max, B\_max | 힘과 자기장 지령의 최대치 |
| CURRENT\_LIMIT | 전류 제한치 |
| int using\_inv\_num | 4코일 시스템인지 8코일 시스템인지를 구분하는 변수. |
| bool gravity\_compensation\_mode | 8코일 시스템에서 중력 보상 항을 힘 지령에 더할 지 결정하는 변수. (4코일 시스템은 항상 더하지 않는다.) |
| bool gravity\_in\_net\_force | 8코일 시스템에서 (위 변수와 상관 없이) 마이크로로봇에 인가된 총 힘을 계산할 때 중력을 더할 지 결정하는 변수. (4코일 시스템은 항상 더하지 않는다.) |
| bool update\_PM\_with\_ref | 마이크로로봇의 위치와 각도를 갱신할 때에 측정 전류가 아닌 지령 전류를 사용하여 계산할 지 결정하는 변수. |
| bool update\_magnetization\_with\_B\_ref | 위 변수가 false 인 경우 힘은 측정 전류를 바탕으로 계산되고 위치가 업데이트 된다. 마이크로로봇이 자기장 지령 방향으로 즉시 정렬 된다고 가정할 지 결정하는 변수. (false일 때 측정 전류를 바탕으로 계산된 자기장 방향으로 마이크로로봇이 정렬된 것으로 가정.) |
| bool fix\_position | 계산된 힘에 상관 없이 마이크로로봇이 제어 영역 중앙에 있다고 가정할 지 결정하는 변수. |
| bool update\_LUT\_with\_Position | 마이크로로봇의 위치를 바탕으로 자기장 LUT를 업데이트할 지 결정하는 변수. false일 경우 ‘coil\_configuration’ 파일들을 지령 계산에 영향을 주지 않으며 오로지 ‘easy\_B\_LUT’ 파일들만 사용하여 전류 지령이 계산된다. |
| bool check\_actuation\_threshold | 4코일 시스템에서 마이크로로봇이 정렬된 방향과 수직한 방향의 힘 지령이 주어진 경우 전류 지령이 발산하게 되는데 이를 방지할 지 결정하는 변수 |
| int BF\_ref\_mode | 8코일 시스템에서 초기 자기장, 힘 지령 |
| int B\_ref\_mode | 4코일 시스템에서 초기 자기장 지령 |
| int F\_ref\_mode | 4코일 시스템에서 초기 힘 지령 |
| float BF\_ref\_frequency | 자기장과 힘 지령에 진동 성분이 있는 경우 해당 지령생성에 사용될 진동수 |
| float BF\_ref\_ang\_vel | 자기장과 힘 지령에 진동 성분이 있는 경우 해당 지령생성에 사용될 각속도 (위 변수에 2 pi 를 곱한 값) |
| int current\_limit\_mode | 전류 지령이 안전치를 넘었을 경우 지령을 수정하는 방법. PROPORTIONAL\_LIMIT\_METHOD 로 설정하면 정렬 방향과 병진 운동의 방향에 왜곡이 없는 보정을 실시한다. |

**<프로그램 실행 전 준비>**

1. 전류 제어 시스템의 통신용 보드와 udp 선 연결.
2. 전류 제어 시스템의 인버터 보드와 코일을 전선으로 연결. 이 때 인버터 보드의 순서와 ‘coil\_configuration’ 혹은 ‘easy\_B\_LUT’ 파일에 기입된 코일의 순서가 같도록 한다.
3. 전류 제어 시스템 전원 공급.

이외에 PC의 Ethernet 포트 ip 설정이 필요한데 이는 <프로그램 로그> 장에서 설명한다.

**<프로그램 실행 방법>**

프로그램을 실행하기에 앞서 RTAI 모듈들을 background에서 실행해 두어야 한다. 실행을 하기위한 명령은 다음과 같으며 PC를 켠 뒤 한 번만 실행하면 된다. (명령은 프로젝트 폴더 내에서 실행한다.)

|  |
| --- |
| sudo sh rtai-insmod.sh |

필요에 의해 RTAI 모듈들을 실행 중지하려는 경우 다음을 실행한다.

|  |
| --- |
| sudo sh rtai-rmmod.sh |

잠시 기다리면 실행이 완료된다. 이후 다음 명령을 통해 프로그램을 실행한다.

|  |
| --- |
| sudo ./microrobot\_pc\_app |

만약 4코일 시스템인지 8코일 시스템인지 variables.cpp 에서 설정을 하지 않았다면 다음과 같이 입력하여 실행하며 설정할 수도 있다.

|  |
| --- |
| sudo ./microrobot\_pc\_app 4  혹은  sudo ./microrobot\_pc\_app 8 |

프로그램 종료는 실행 도중 0+enter를 입력하면 된다.

전류 제어 시스템의 통신용 보드가 사전 코딩된 설정에 따라 프로그램 구동 준비 시간이 필요할 수 있다. ‘programmed for 4 inverter boards’ 라고 기재된 통신용 보드를 4코일 시스템에 사용하는 경우는 프로그램 시작 즉시 지령 전달 및 수행이 가능하다.

**<프로그램 로그>**

마이크로로봇 제어 프로그램은 (current\_result.csv 파일가 별도로) 실행 중 콘솔에 로그를 출력하도록 되어있다. 프로그램 구동 중 에러가 있는 경우엔 출력과 함께 프로그램이 자동 종료 된다. 프로그램이 정상 구동 된 뒤에는 1초마다 다음의 로그를 출력한다.

[t]s : Mode [m], Status [s]

위 로그는 실행 시작 후 [t]초가 된 때에 출력된 것으로 현재 프로그램이 모드 m과 상태 s에 있음을 의미한다. 모드와 상태에 관련된 표는 다음과 같다.

|  |  |  |
| --- | --- | --- |
| 모드 [m] | 1 | INIT mode : 통신 및 인버터 보드의 상태가 정상인지 확인하는 모드 |
| 2 or 3 or 4 | RUN mode : 주어진 자기장과 힘 지령을 수행하는 모드 |
| 상태 [s] | 1 | 상태 확인(점검) 중 / 구동 중 |
| 2 | 정상 |
| 3 | 통신용 보드와 연결 안됨 |
| 4 | 통신용 보드와 통신은 되나 데이터가 잘못 수신됨 |
| 5 | 통신용 보드와 인버터 사이의 SPI 통신선 연결 문제 |
| 6 | 통신용 보드와 인버터 사이의 SPI 통신선 데이터 문제 |
| 7 | PC 위의 총괄 제어 시스템과의 통신 문제 |
| 8 | 기타 에러 |
| 9 | 인버터의 전류 혹은 전압에서 위험을 감지 |

(현재 7번 상태는 사용되지 않음)

PC와 통신용 보드를 연결한 뒤 전류 제어 시스템의 전원을 넣고 프로그램을 실행한다. 프로그램이 바로 종료되지 않았다면 다음과 같이 로그가 출력이 되어야 정상적으로 구동된 것이다.

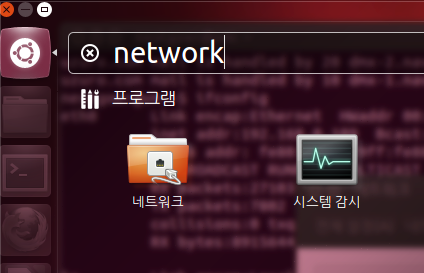
|  |
| --- |
| RT timer started  RT setting done  B LUT is loaded.  Server is ready!  0s : Mode 1, Status 1  1s : Mode 1, Status 2  2s : Mode 1, Status 2  3s : Mode 1, Status 2  … |

만약 통신용 보드에 ‘programmed for 4 inverter boards’가 기재되어 있고 프로그램을 8코일 시스템에서 구동한다면 10초 이상 기다리며 status가 2로 바뀌는 지 확인한다. 통신용 보드와 udp 선이 잘 연결되지 않았거나 PC의 이더넷 포트 ip가 설정이 잘못 되었다면 다음과 같은 로그가 출력된다.

|  |
| --- |
| RT timer started  RT setting done  B LUT is loaded.  Server is ready!  0s : Mode 1, Status 1  1000 connection lost in a row. Escape.  1s : Mode 1, Status 2  2s : Mode 1, Status 2  3s : Mode 1, Status 2  … |

[PC의 이더넷 포트 ip 설정 (ubuntu desktop)]

우분투 데스크탑 버전에서는 네트워크 매니저라는 패키지가 그 어떤 네트워크 설정보다 우선순위가 높다. /etc/resolv.conf 나 /etc/network/interfaces를 직접 수정하더라도 그것이 반영되지 않고 네트워크 매니저에 지정되어 있는 값이 적용된다. 대쉬홈에서 ‘network’나 ‘네트워크’로 검색하여 네트워크 도구를 찾아 실행한다.



유선 연결을 선택한 뒤 수정이 가능한 경우 다음으로 수동 고정 ip를 다음의 값으로 설정한다.

address : 192.168.100.100

netmask : 255.255.255.0

gateway : 192.168.100.1

만약 유선 설정의 변경이 네트워크 도구에서 불가능 하다면 /etc/network/interfaces 의 내용에 다음을 추가한다.

|  |
| --- |
| auto eth0  iface eth0 inet static  address 192.168.100.100  netmast 255.255.255.0  gateway 192.168.100.1 |

수정하여 저장한 뒤 다음을 실행하여 설정을 적용한다.

|  |
| --- |
| sudo service network-manager restart |

이 작업 이후에도 구동 후 통신이 되지 않는다면 /etc/network/interfaces 의 내용에서 eth0 을 eth1로 바꾸어 저장한 뒤 다시 ‘sudo service network-manager restart’ 로 설정을 적용하고 프로그램을 구동해본다.

정상적으로 통신이 되어 실험을 하던 도중 통신이 갑자기 끊긴 경우에도 ‘1000 connection lost in a row. Escape.’ 문구가 나타날 수 있다. 이 경우에는 자동으로 통신 복구를 시도하도록 되어있으나 자동으로 복구가 안 될 경우 프로그램을 종료한 뒤 다시 실행해본다.

**<프로그램 실행 후 조작 방법>**

프로그램 실행 후 터미널 콘솔에 키보드를 통해 값을 넣고 엔터를 입력하여 조작을 한다. (입력창이 따로 뜨지 않고 화면 출력이 계속 있는데 그냥 입력하면 된다.)

[모드 변경]

|  |  |
| --- | --- |
| 1 + enter | INIT mode로 변경 |
| 2 + enter | RUN mode로 변경 |
| 0 + enter | 프로그램을 종료한다. |

[지령 변경]

설정된 지령은 RUN mode에 있을 때에 수행된다. INIT mode에서 자기장 지령이나 힘 지령을 변경 시키면 바로 실행되지 않는다. RUN mode에서 자기장 지령이나 힘 지령을 변경하면 바로 변경되어 수행한다.

|  |  |
| --- | --- |
| b + enter | 수행할 수 있는 자기장 지령의 종류를 표시한다. 나타난 보기의 번호와 enter를 입력하면 수행 자기장 지령이 변경된다. |
| f + enter | 수행할 수 있는 힘 지령의 종류를 표시한다. 나타난 보기의 번호와 enter를 입력하면 수행 힘 지령이 변경된다. |

4코일 시스템의 경우 x축에 정렬된 자석에는 y축 힘을 인가할 수 없고, y축에 정렬된 자석에는 x축 힘을 인가할 수 없어 자기장과 힘 지령이 동시에 수행될 수 있도록 지령을 만들어 두었다. 자기장과 힘 지령이 한 번에 수정됨 (b+enter와 f+enter의 기능이 같음)

[기타 조작]

|  |  |
| --- | --- |
| n + enter | 제어할 전류(인버터 보드)의 수. 개수를 입력 후 enter (중대형시스템 : 8, 소형시스템 : 4) / 실행 중 인버터 보드의 수를 변경할 경우 인버터 보드들의 리셋이 반드시 필요하므로 입력 후 10초간 대기한다. |
| w + enter | 자기장이나 힘 지령이 진동하는 함수일 경우 해당 함수의 주파수 변경 / 원하는 주파수를 float 형으로 입력한 뒤 enter |
| m + enter | 전류 제한 방법 변경 / proportional limit을 사용하는 경우 이동 방향이나 정렬 방향에 왜곡이 없다. |

n + enter : 제어할 전류(인버터 보드)의 수 / 개수를 입력 후 enter (중대형시스템 : 8, 소형시스템 : 4)

m + enter : 전류 제한 방법 변경 / proportional limit을 사용하는 경우 이동 방향이나 정렬 방향에 왜곡이 없다.

w + enter : 자기장이나 힘 지령이 진동하는 함수일 경우 해당 함수의 주파수 변경 / 원하는 주파수를 float 형으로 입력한뒤 enter.

**<프로그램 종료>**

전류 제어 시스템의 통신용 보드는 PC와의 통신이 끊어진 뒤 1초간 통신을 기다린다. 만약 RUN모드에서 통신이 1초동안 오지 않는다면 INIT모드로 돌아가 전류 출력을 하지 않게 된다. 반대로 볼 때, 지령 수행 중 안정적인 프로그램 종료를 위해서는 ‘1 + enter’를 입력하여 INIT 모드로 설정하여 전류 출력을 중지 한 뒤 ‘0 + enter’로 종료하는 것이 좋다.