Human-centered
Robotics & Vision Lab.

LinearActuator(pololu)

2019.06.13

김영기
Dept. of Electronics, Information & Communications
Daejeon University(HRVL)

jrk Configuration utility initialize

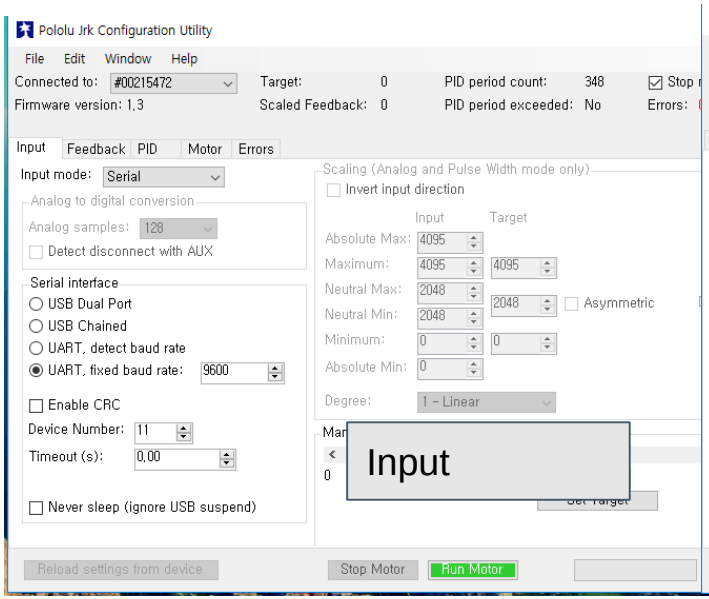
- pololu의 jrk Configuration Utility 프로그램을 설치 한 후 (설치링크 :https://www.pololu.com/docs/0J38/3.a)
- https://www.pololu.com/product/2327 링크에서 다음 순서에 따라서 jrk configuration utility 프로그램을 셋팅해준다.

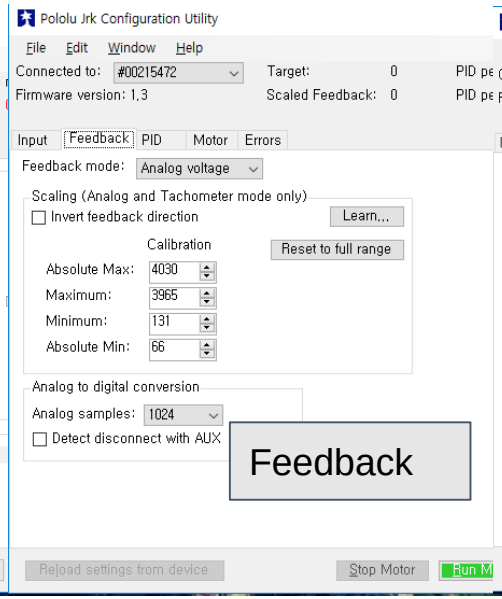
1. If you have not already, read through the Jrk G2 Motor Controllers User's Guide and download its drivers and configuration software.
2. Before connecting power and your actuator to your Jrk, confirm that it is working by connecting your Jrk to a PC with a USB cable and launch the configuration utility. The red LED should be on, and the green LED should be flickering quickly.
3. Download the appropriate settings file for your linear actuator:
    - for versions with the **5:1** gear ratio: Jrk 21v3 settings file for use with LACTxP-12V-5 (2k txt)
    - for versions with **10:1 or 20:1** gear ratios: Jrk 21v3 settings file for use with LACTxP-12V-10 or LACTxP-12V-20 (2k txt)

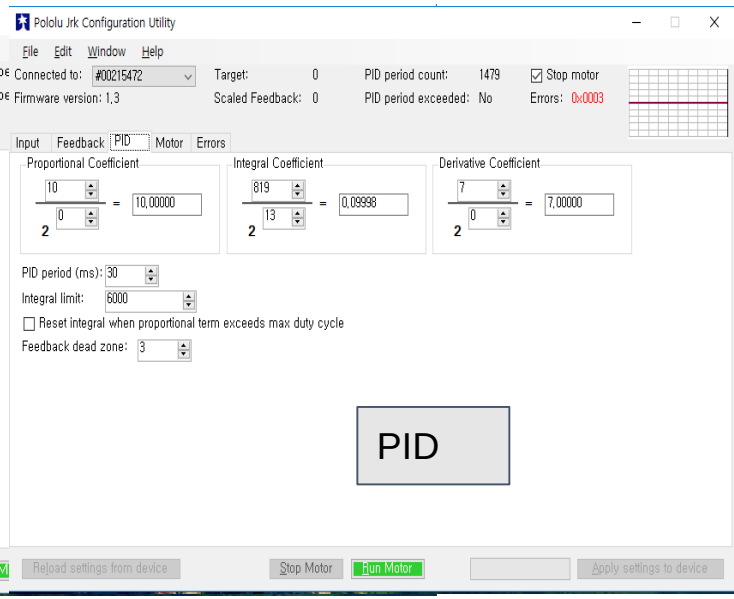
Note: These same settings files will work also with the Jrk G2 24v13 if you open them with a text editor and change the "product" field near the top from "21v3" to "24v13".

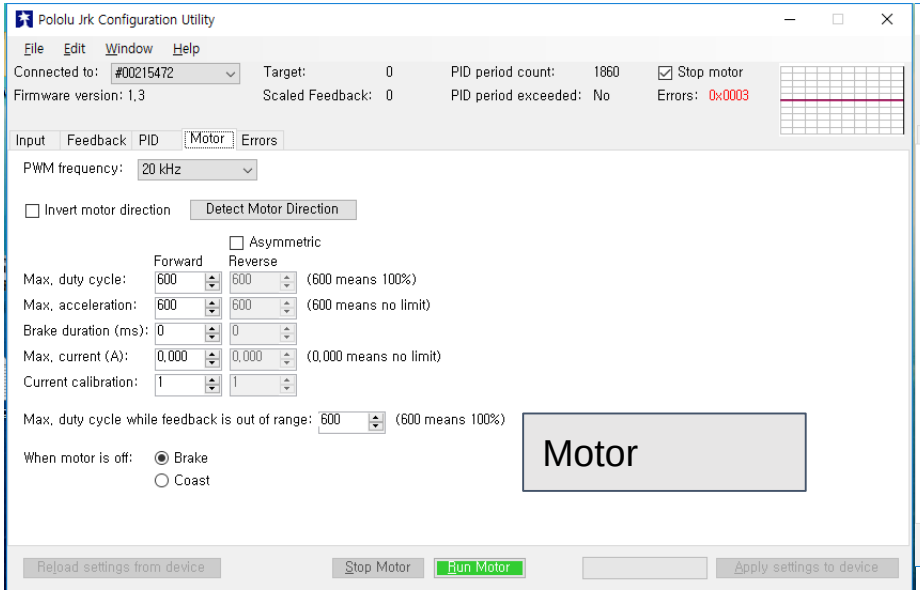
4. In the configuration utility, choose File → Open settings file (Ctrl + O), and navigate to the location of the settings file you downloaded in step 3.
5. Click on the PID tab of the configuration utility and verify that the proportional and derivative coefficients are not zero. If they are zero, the settings file was probably not loaded properly and you should try performing the previous step again.
6. Click "Apply settings".
7. With your power supply off and USB disconnected, connect your linear actuator to your Jrk using the connections shown in the picture above.
8. Turn on power, plug in USB, and reconnect to the configuration utility (use the "Connected to" drop down box if the configuration utility doesn't automatically reconnect to your Jrk).
9. On the Status tab, move the slider around to change the target position and get your actuator to move the target position.
10. The settings in these files should work fairly well with any length Glideforce light-duty actuator that has a feedback potentiometer (model LACTx**P**-12V). However, to ensure you can control your actuator across its full stroke, you should recalibrate the feedback. Instructions for doing this can be found in the analog feedback section of the Jrk G2 User's Guide.

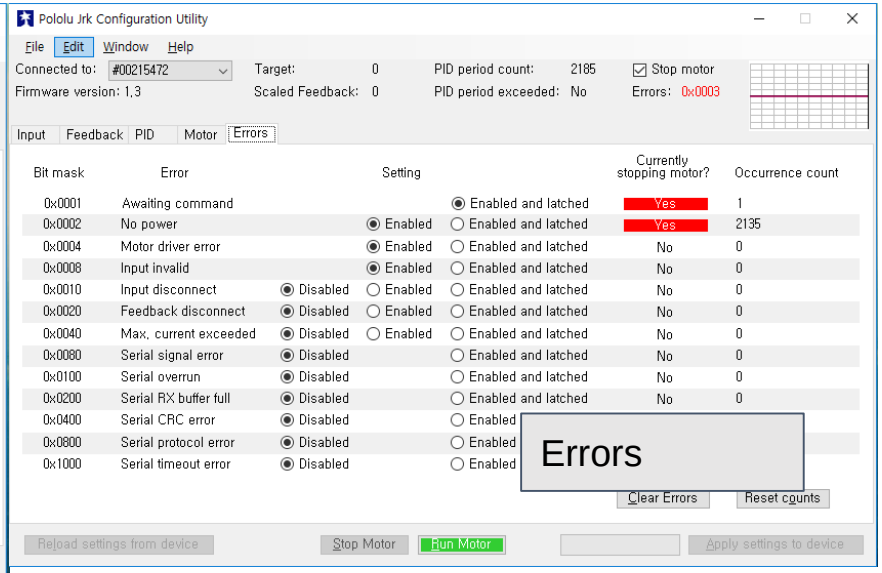
jrk Configuration utility initialize











jrk Motor control command

- jrk Guide를 참고하여 아두이노 코드 작성 https://www.pololu.com/docs/0J38
- jrk Git내용을 참고하여 lib 사용 https://github.com/pololu/jrk-g2-arduino

Motor Off

Compact protocol: **0xFF**
Pololu protocol: **0xAA, device number, 0x7F**

This command will turn the motor off by setting the Awaiting Command error bit. The jrk will not restart the motor until it receives a Set Target command. The jrk can be configured to either brake or coast while the motor is off (Section 3.e).

Set Target High Resolution

Compact protocol, binary: **110LLLLL, 0HHHHHHH**
Compact protocol, hex: **0xC0 + target low 5 bits, target high 7 bits**
Pololu protocol, binary: **10101010, device number, 010LLLLL, 0HHHHHHH**
Pololu protocol, hex: **0xAA, device number, 0x40 + target low 5 bits, target high 7 bits**
(where target is the 12-bit number HHHHHHHLLLLL)

This command clears the Awaiting Command error bit and (if Input Mode is Serial) lets you set the 12-bit target to any of its allowed values (0–4095). The meaning of the target depends on what Feedback Mode the jrk is in (Section 3.c). The lower 5 bits of the command byte represent the lower 5 bits of the target, while the lower 7 bits of the data byte represent the upper 7 bits of the target.

For example, if you want to set the target to 3229 (110010011101 in binary), you could send the following byte sequence:

in binary: 11011101, 01100100
in hex: 0xDD, 0x64
in decimal: 221, 100

Here is some example C code that will generate the correct serial bytes, given an integer “target” that holds the desired target (0–4095) and an array called serialBytes:

```
serialBytes[0] = 0xC0 + (target & 0x1F); // Command byte holds the lower 5 bits of target.
serialBytes[1] = (target >> 5) & 0x7F;   // Data byte holds the upper 7 bits of target.
```