EZ Controller (IR-CT01)
User Manual

ROBOT

INDEX

1 Overview

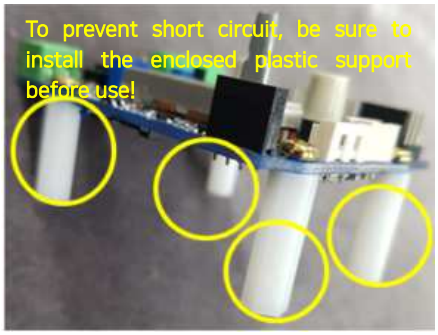
Arduino-based EZ Controller is a controller to easily operate / test mightyZAP linear servo actuators.
The features of EZ Controller are as follows.

- Operate the linear actuator simply and easily through various input devices on the board by using the built-in basic program.
- Various motion programming is available by example program provided from us or by your own coding via Arduino.
- Connectable with various external accessories. (External switch, etc.)
- No need for a separate circuitry to control the mightyZAP linear actuator. Also, for safety, it is equipped with power protection circuits for prevention of reverse power, over-current, and static electricity.

1.1 Precautions

The following precautions require special attention when using, so be sure to read them carefully. Please note that warranty will be void for problems caused by failure to comply with the following.

1. Do not apply excessive force when connecting products. Also, do not apply excessive force to the components or the pins on the board. It may cause malfunction.
2. Be advised that it may cause a short circuit when connecting power to the board on the metal plate that is not insulated. Be sure to attach the enclosed plastic support to the board before use. (See the picture on the right)
3. Since the voltage applied to the EZ controller is directly applied to the connected actuator, check the operating voltage range of the actuator and check the polarity when connecting external power.
   Ex) If user connects the actuator whose input voltage is 7.4V (D7 or L7 series) to the board, and accidently connects 12V power to the board may cause the damage of actuator.



4. When connecting the actuator to the board, make sure to check the polarity of the pins.
   It may cause motor or actuator PC board damage.
5. Use the appropriate connector to prevent damage due to incorrect insertion.
6. Apply proper input to each terminal.
   Make sure proper signal voltage and pay attention to the proper wiring between power and signal terminal.
7. Do not overturn the variable resistance knob when setting the start and end points of the rod.
8. Keep the product away from fire, water, dust and oil and keep the product out of reach of children.
9. This product is designed for indoor use. Please refrain from using outdoors.

1.2 Proper Storage

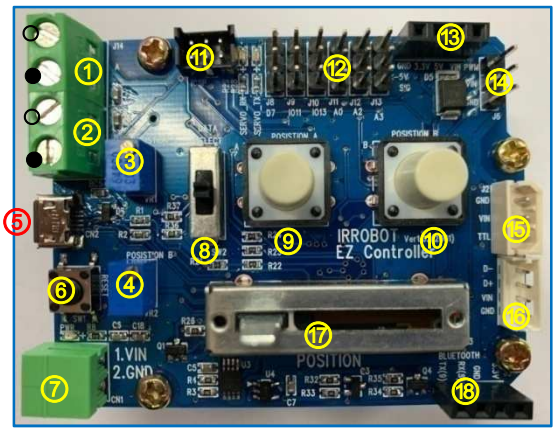
Do not use or store the product in the following extreme environments. It may cause malfunction or product damage.

- High temperature environment over 60℃ or low temperature below minus 20℃
- Direct sunlight or near fire
- Hot, humid and dusty places
- Severe vibration condition
- Places that can cause static electricity

2 Function & Operation

2.1 Function

■ EZ Controller Components and Description



① A Position External switch input (○: +, ●:-)

② B Position External switch input (○: +, ●:-)

: Position command by external switch or external signal. It is same function as ⑨,⑩ push button.

: For position setting, use ③ and ④

③ "A" Position setting V/R : "A" position setting by adjusting variable resistor. (A position command made by ① or ⑨)

④ "B" Position setting V/R : "B" position setting by adjusting variable resistor. (B position command made by ② or ⑩)

Set the positions of the A and B points by adjusting the blue V/R ③ (A position) and ④ (B position) respectively.
-Clockwise:- (actuator retraction direction)
-Counterclockwise: + (actuator extension direction)

For convenience, it is recommended to set A to minimum position and B to maximum position as shown below.
-A point (part ③): Turn clockwise to set the minimum position
-B point (part ④): Turn counterclockwise to set the maximum position
Please note that the minimum and maximum positions of A and B may be reversed and changed depending on user settings.

⑤ PC connection micro USB terminal : Arduino Sketch download/Serial communication

⑥ Reset Switch: Controller reset

⑦ 12V main power input terminal

⑧ Mode selection Switch

⑨ A Position push button Switch

⑩ B Position push button Switch

⑪ MCU F/W download connector(ICSP) : User manipulation prohibited

⑫ Arduino I/O 핀(Digital:3 , Analog:3 / GND, 12V, SIG)

⑬ Internal power terminal(GND / 3.3V / 5V / 12V) : power output terminal internally used.
[Current Limit of each terminal]
- 3.3V : ~150mA / 5V : ~ 900 mA / 12V : Depending on Input Voltage Source

⑭ PWM port

⑮ TTL port

⑯ RS-485 port

⑰ Linear Potentiometer for manual position

⑱ User external communication terminal (Bluetooth : TX / RX / GND / 3.3V)

2.2 Mode Selection

- Test the operation of the actuator using the basic operation program built into the EZ Controller.
- The basic operation program consists of Manual Mode, Position Select Mode, and Position Toggle Mode. (The mode switch only affects the basic operation program.)

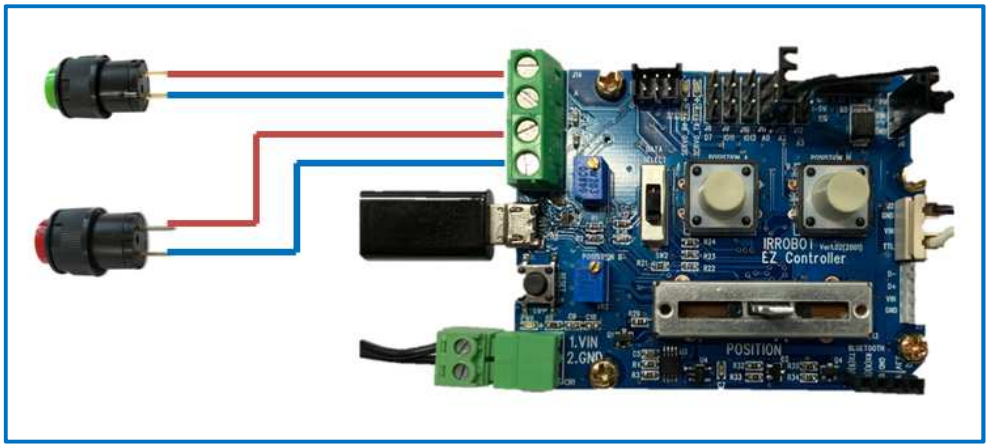
Manual	Position Select	Position Toggle
		

- **Manual Mode** : Connect the actuator, move the Mode switch(⑧) to the lowest position, set it to Manual Mode. Then, operate actuator by change the position of the linear potentiometer(⑰).
- **Position Select Mode :** Connect the actuator, move the Mode switch(⑧) to the middle position, set it to Position Select Mode. Adjust the blue position setting variable resistors (A:③ &B:④) to set A and B Position respectively, and then press the white push buttons(A:⑨& B:⑩) in the center to operate the actuator as Point A and Point B.
- **Position Toggle Mode :** Connect the actuator, move the Mode switch⑧) to the top, set it to Position Toggle Mode. Adjust the blue position setting variable resistors (A:③ &B:④) to set A and B Position respectively, and then press any of the white push buttons(⑨or⑩) in the center to operate the actuator to point A and B.

2.3 External Switch

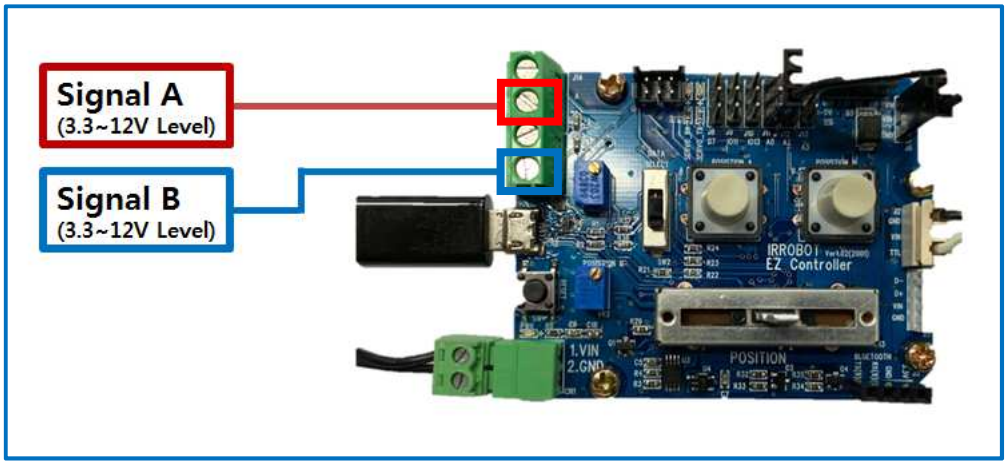
■ **A/B position external switch input & External signal input** (Pic. ①,②)
Connect external switches or apply a voltage level signal for position command. It is the same function as the white push button switches (⑨, ⑩). Only the position command can be made, and each position setting should be made through the blue variable resistors (③ and ④).

1) **External Switch Connection**
: Connect the both ends of the terminal with switches to input the signal. (Short circuit at both ends of the terminal)



2) External Signal Input

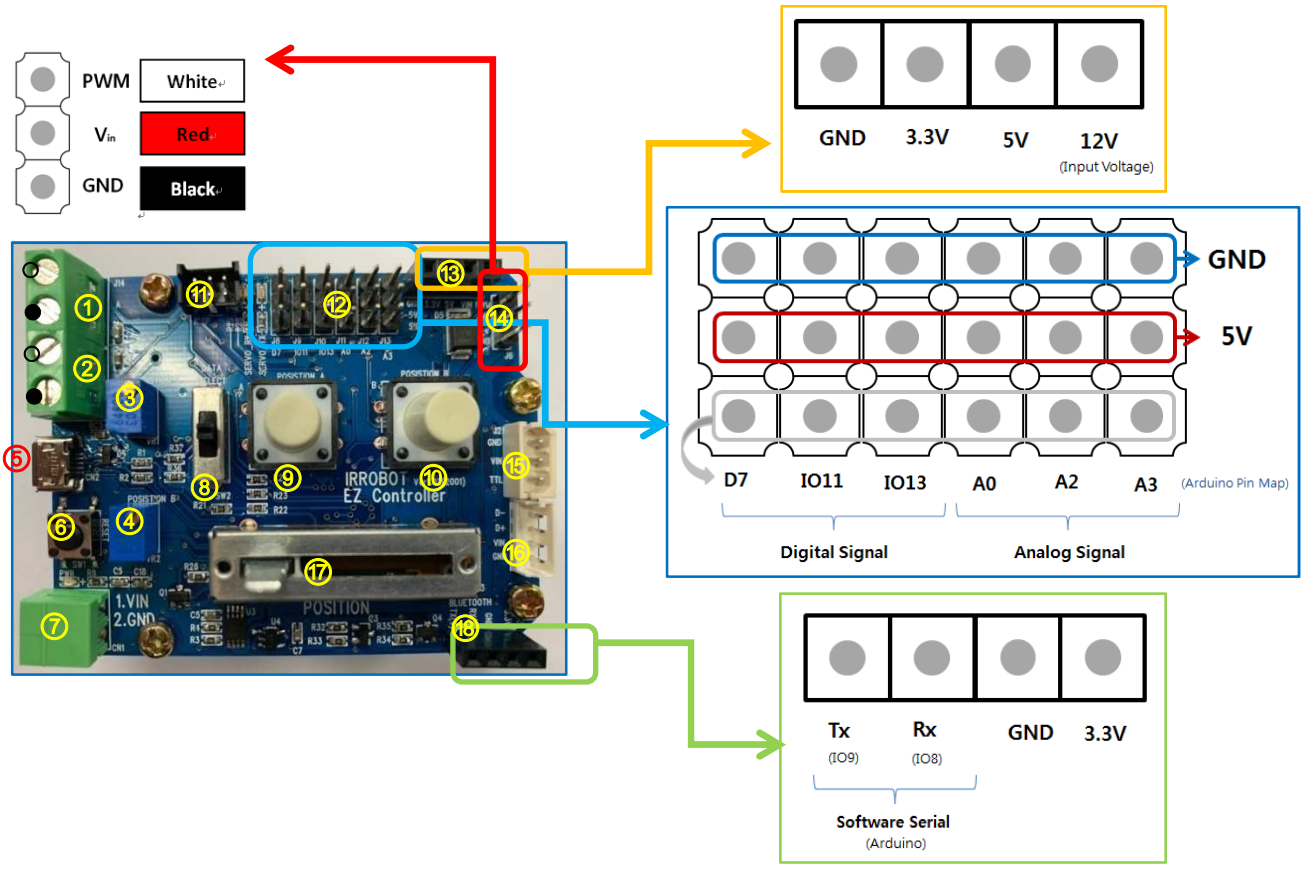
: To recognize the signal in the second hole of each terminal, apply 3.3~12V level signal.



2.4 Part Pinmap

- Parts No : Part number described on the manual Page 4.
- Pin Name : Arduino(Leonardo) I/O Pin

Part No	Pin Name	Purpose	Part No	Pin Name	Purpose
①,⑨	I/O10	GPIO- A Signal	⑫	A0,A2,A3	GPIO - Analog
②,⑩	D3	GPIO- B Signal	⑭	D5	GPIO - PWM
③	A4	Analog-A Variable resistor	⑮,⑯	D0,D1	For TTL,RS485 Communication
④	A5	Analog-B Variable resistor	⑰	A1	Analog -Potentiometer
⑤	Serial	USB Serial, Sketch download	⑱	IO8,IO9	Software Serial
⑧	D4,IO12,D6	GPIO – Slide SW			
⑫	D7,IO11,IO13	GPIO - Digital			



3 Example of Actuator Control through Arduino IDE

Actuator control via the Arduino IDE's data communication protocol (RS-485 or TTL). This is for the advanced users of Arduino who want to control actuators more freely through data communication of Arduino.

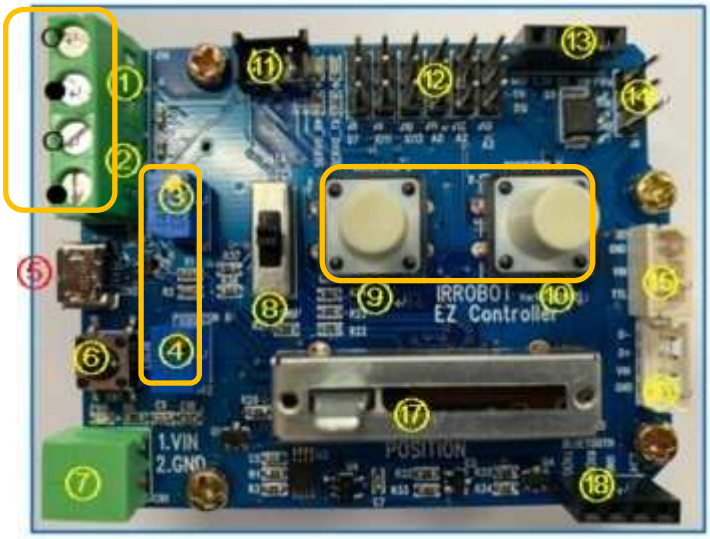
3.1 Overview

Here are Arduino examples to control mightyZAP through EZ Controller. The examples are for serial data communication (TTL or RS-485 communication) and do not support PWM communication.
The EZ Controller is designed based on Arduino Leonardo, and the Arduino API provided by us is based on Arduino Leonardo / Uno. For detailed description of each parameter function, please refer to the mightyZAP user manual.

3.2. Example - 2 Position Control (RS-485/TTL)

The EZ controller contains two variable resistors (Figures 3 and 4) that can assign two positional points and two buttons (Figures 9 and 10) that commands to reach the assigned points. (Alternatively, it is also possible to reach the assigned points with an external physical switch or external voltage signal (Figure 1 or 2). For detailed information about external switch, see 2.3 External switches on page5.

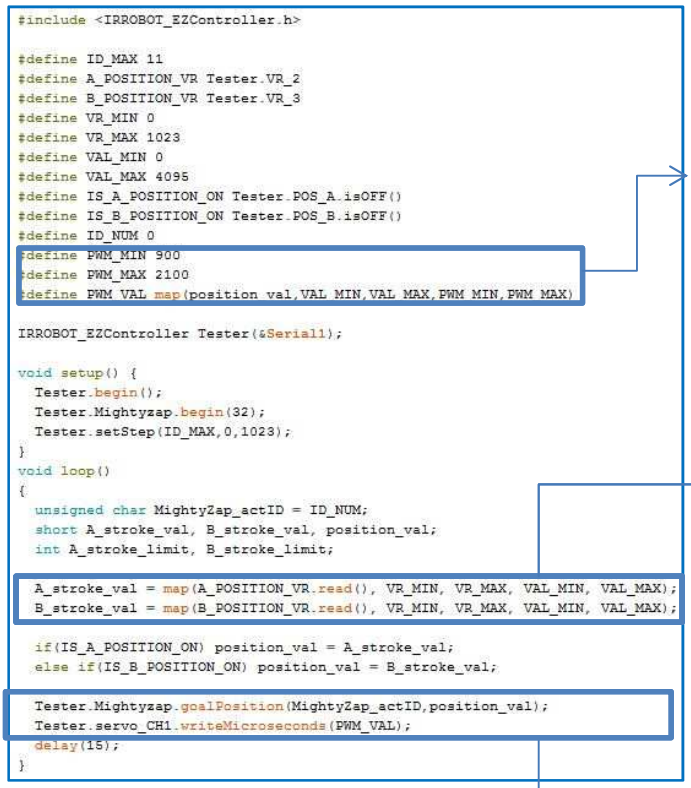
Here is the example of moving to 2 assigned positions which is set by position setting variable resistors (#3&4) through 2 switches (#9&10). Select [Example] - [IRROBOT_EZ Controller] - [EZ]-[EZ Controller_2Positions]



[Description]

- Mode switch (#8) can be positioned anywhere. (Mode switch only works in the built-in basic program)
- Supply power to the input power terminal(#7). Make sure correct input voltage(7.4V or 12V) and correct polarity.(GND and VCC)
- Carefully insert the connector suitable for the selected communication. (#14 PWM / #15 TTL / #16 RS-485)
  (For PWM connector (#14), refer to the page 6 to make sure correct polarity.)
- Set two positional points (position A & B) by turning variable resistors (#3&4). (Clockwise turn – Retracting direction(short stroke) / Counter clockwise turn- Extension direction(long stroke))
- Press A, B button (#9, 10) to make positional command to the assigned goal positions.
- It is also possible to reach the assigned points with an external switch or external voltage signal (#1 or 2). For detailed information about external switch, see 2.3 External switches on page5.

- [Program Description]



```
#include <IRROBOT_EZController.h>

#define ID_MAX 11
#define A_POSITION_VR Tester.VR_2
#define B_POSITION_VR Tester.VR_3
#define VR_MIN 0
#define VR_MAX 1023
#define VAL_MIN 0
#define VAL_MAX 4095
#define IS_A_POSITION_ON Tester.POS_A.isOFF()
#define IS_B_POSITION_ON Tester.POS_B.isOFF()
#define ID_NUM 0
#define PWM_MIN 900
#define PWM_MAX 2100
#define PWM_VAL map(position_val,VAL_MIN,VAL_MAX,PWM_MIN,PWM_MAX)

IRROBOT_EZController Tester(&Serial1);

void setup() {
  Tester.begin();
  Tester.Mightyzap.begin(32);
  Tester.setStep(ID_MAX,0,1023);
}
void loop()
{
  unsigned char MightyZap_actID = ID_NUM;
  short A_stroke_val, B_stroke_val, position_val;
  int A_stroke_limit, B_stroke_limit;

  A_stroke_val = map(A_POSITION_VR.read(), VR_MIN, VR_MAX, VAL_MIN, VAL_MAX);
  B_stroke_val = map(B_POSITION_VR.read(), VR_MIN, VR_MAX, VAL_MIN, VAL_MAX);

  if(IS_A_POSITION_ON) position_val = A_stroke_val;
  else if(IS_B_POSITION_ON) position_val = B_stroke_val;

  Tester.Mightyzap.goalPosition(MightyZap_actID,position_val);
  Tester.servo_CH1.writeMicroseconds(PWM_VAL);
  delay(15);
}
```

Example of setting two points with the value of a variable resistor and moving them to the corresponding point when the buttons of A and B are pressed.

Mapping for PWM value
Short : 900 Long : 2100

Read the variable resistor values of Position A and Position B and assign values to “A_stroke_val” and “B_stroke_val” variables respectively.

The map () function maps the variable resistor's resistor value to the actuator's position range.

* In case not using a variable resistor,,

A_stroke_val = 100;
B_stroke_val = 3600;

It has a scale of 4096 and enters a value between 0 and 4096.

Tester.MightyZAP.goalPosition() : TTL,485 control
Tester.servof_CH1.writeMicroseconds() : PWM control

3.3. Example - TogglePosition

This is an example of inverting 2 Positions designated by 1 toggling button.
Select [Example] - [IRROBOT_EZController] - [EZ]- [EasyControl_TogglePosition]

[Description]

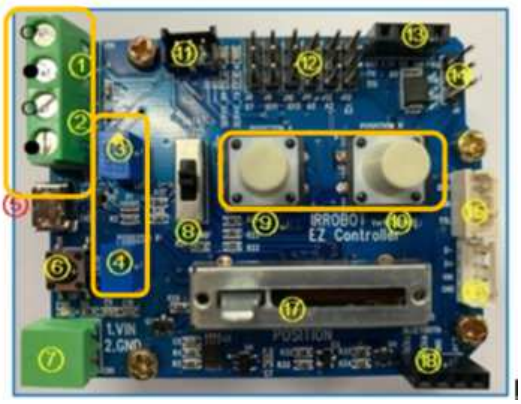
- Mode switch (#8) can be positioned anywhere. (Mode switch only works in the built-in basic program)
- Supply power to the input power terminal (#7). Make sure correct input voltage(7.4V or 12V) and correct polarity.(GND and VCC)
- Carefully insert the connector suitable for the selected communication. (#14 PWM / #15 TTL / #16 RS-485)
  (For PWM connector (#14), refer to the page 6 to make sure correct polarity.)
- Set two positional points (position A & B) by turning variable resistors (#3&4). (Clockwise turn – Retracting direction(short stroke) / Counter clockwise turn- Extension direction(long stroke))
- Press A or B button (#9 or 10) to make positional command to the assigned goal positions. (Either button between A and B)
- It is also possible to reach the assigned points with an external switch or external voltage signal (#1 or 2). For detailed information about external switch, see 2.3 External switches on page5.

[Program Description]

```
#include <IRROBOT_EZController.h>

#define ID_MAX 11
#define A_POSITION_VR Tester.VR_2
#define B_POSITION_VR Tester.VR_3
#define VR_MIN 0
#define VR_MAX 1023
#define VAL_MIN 0
#define VAL_MAX 4095
#define IS_A_POSITION_ON Tester.POS_A.isOFF()
#define IS_B_POSITION_ON Tester.POS_B.isOFF()
#define ID_NUM 0
#define PWM_MIN 900
#define PWM_MAX 2100
#define PWM_VAL map(position_val,VAL_MIN,VAL_MAX,PWM_MIN,PWM_MAX)

IRROBOT_EZController Tester(&Serial1);

short position_val;
bool tg_flag;

void setup() {
  Tester.begin();
  Tester.Mightyzap.begin(32);
  Tester.setStep(ID_MAX,0,1023);
}
void loop() {
  unsigned char MightyZap_actID = ID_NUM;
  short A_stroke_val;
  short B_stroke_val;

  A_stroke_val = map(A_POSITION_VR.read(),VR_MIN,VR_MAX,VAL_MIN,VAL_MAX);
  B_stroke_val = map(B_POSITION_VR.read(),VR_MIN,VR_MAX,VAL_MIN,VAL_MAX);

  if(IS_A_POSITION_ON || IS_B_POSITION_ON) tg_flag ^= 1;

  if(tg_flag) position_val = A_stroke_val;
  else position_val = B_stroke_val;

  Tester.Mightyzap.goalPosition(MightyZap_actID,position_val);
  Tester.servo_CH1.writeMicroseconds(PWM_VAL);
  delay(15);
}
```

Example in which the two points set as variable resistors are reversed from point A to point B and point B to point A when the A or B button is pressed.

When the A or B button is pressed, the tg_flag value declared as bool is inverted, and when the value is 1, it moves to the A point, and when it is 0, it moves to the B point.



3.4. Example –Manual Position

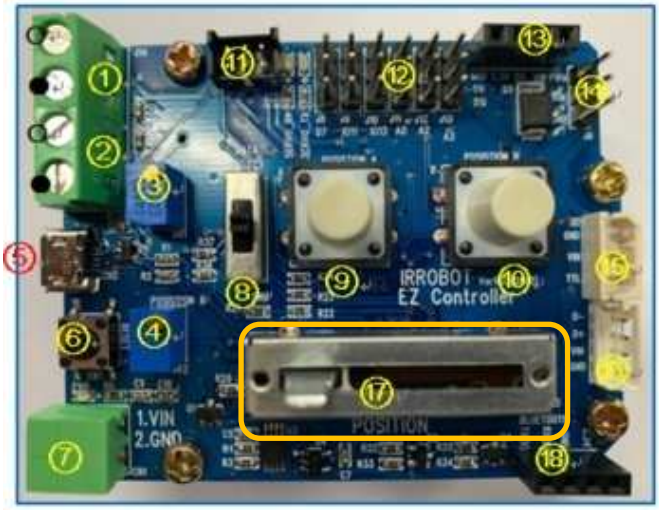
Moving the position of the actuator by moving distance of the linear potentiometer.
Select [Example] - [IRROBOT_EZController] - [EZ]– [EasyControl_MPosition]

[Description]

- Mode switch (#8) can be positioned anywhere. (Mode switch only works in the built-in basic program)
- Supply power to the input power terminal (#7). Make sure correct input voltage(7.4V or 12V) and correct polarity.(GND and VCC)
- Carefully insert the connector suitable for the selected communication. (#14 PWM / #15 TTL / #16 RS-485)
  (For PWM connector (#14), refer to the page 6 to make sure correct polarity.)
- By adjusting the variable linear potentiometer (#17), user is able to operate actuator. In the manual mode, it is not affected by the setting of the position setting variable resistor (# 3 & 4), and as the linear potentiometer moves, the actuator position can be controlled in the full stroke section.

[Program Description]

```
#include <IRROBOT_EZController.h>

#define ID_MAX 11
#define MANUAL_POSITION_VR Tester.VR_1
#define VR_MIN 0
#define VR_MAX 1023
#define VAL_MIN 0
#define VAL_MAX 4095
#define ID_NUM 0
#define PWM_MIN 900
#define PWM_MAX 2100
#define PWM_VAL map(position_val,VAL_MIN,VAL_MAX,PWM_MIN,PWM_MAX)

IRROBOT_EZController Tester(&Serial1);

short position_val;

void setup() {
  Tester.begin();
  Tester.Mightyzap.begin(32);
  Tester.setStep(ID_MAX,0,1023);
}
void loop()
{
  unsigned char MightyZap_actID = ID_NUM;
  short Manual_position_val;
  short A_stroke_val;
  short B_stroke_val;

  Manual_position_val  = map(MANUAL_POSITION_VR.read(),VR_MIN,VR_MAX,VAL_MIN,VAL_MAX);
  position_val = Manual_position_val;
  Tester.Mightyzap.goalPosition(MightyZap_actID,position_val);
  Tester.servo_CH1.writeMicroseconds(PWM_VAL);
  delay(15);
}
```



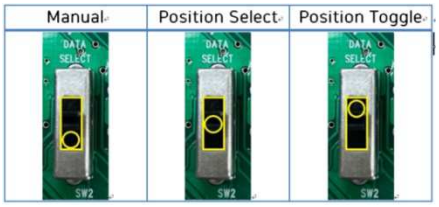
Actuator position operation after reading
Potentiometer value with “Manual_position_val “variable

3.5. Example –Basic Function

Switching control mode and controlling actuator through Mode selection slide switch.
Select [Example] - [IRROBOT_EZController] - [EZ]-[EasyControl_BasicFunction] 선택

[Description]

- Supply power to the input power terminal (#7). Make sure correct input voltage(7.4V or 12V) and correct polarity.(GND and VCC)
- Carefully insert the connector suitable for the selected communication. (#14 PWM / #15 TTL / #16 RS-485)
  (For PWM connector (#14), refer to the page 6 to make sure correct polarity.)
- Switch the mode with the mode slide switch (#8) to operate the actuator.
  1) Mode0 : Manual Control (Switch Bottom)
  2) Mode1 : 2 Position Control (Switch Center)
  3) Mode2 : Toggle Control (Switch Top)
- When operating in Mode1 (Manual Control), note that the position setting variable resistor values by # 3 and 4 are set to the limit value of the movable range of the variable linear potentiometer (# 17).



[Program Description]


```
if(short_stroke_limit>long_stroke_limit)
{
  int temp = short_stroke_limit;
  short_stroke_limit = long_stroke_limit;
  long_stroke_limit = temp;
}
if(IS_MANUAL_MODE_ON)
{
  if(Manual_position_val<short_stroke_limit) Manual_position_val = short_stroke_limit;
  else if(Manual_position_val>long_stroke_limit) Manual_position_val = long_stroke_limit;
  position_val = Manual_position_val;
}
else if(IS_2P_MODE_ON)
{
  if(IS_A_POSITION_ON) position_val = A_stroke_val;
  else if(IS_B_POSITION_ON) position_val = B_stroke_val;
}
else if(IS_TOGGLE_MODE_ON)
{
  if(IS_A_POSITION_ON || IS_B_POSITION_ON){
    if(!Sw_status){
      if(sw_cnt++>7){
        tg_flag ^= 1;
        Sw_status = 1;
        sw_cnt = 0;
      }
    }
    else sw_cnt = 0;
  }
  else {
    sw_cnt = 0;
    if(!IS_A_POSITION_ON && !IS_B_POSITION_ON){
      if(cnt++>7){
        cnt = 0;
        Sw_status = 0;
      }
    }
  }
  if(tg_flag ==1) position_val = A_stroke_val;
  else position_val = B_stroke_val;
}
Tester.Mightyzap.goalPosition(MightyZap_actID,position_val);
Tester.servo_CH1.writeMicroseconds(PWM_VAL);
delay(10);
}
```

```
#include <IRROBOT_EZController.h>

#define ID_MAX 11
#define MANUAL_POSITION_VR Tester.VR_1
#define A_POSITION_VR Tester.VR_2
#define B_POSITION_VR Tester.VR_3
#define IS_MANUAL_MODE_ON Tester.MODE_0.isOFF()
#define IS_2P_MODE_ON Tester.MODE_1.isOFF()
#define IS_TOGGLE_MODE_ON Tester.MODE_2.isOFF()
#define IS_A_POSITION_ON Tester.POS_A.isOFF()
#define IS_B_POSITION_ON Tester.POS_B.isOFF()

#define VR_MIN 0
#define VR_MAX 1023
#define VAL_MIN 0
#define VAL_MAX 4095
#define ID_NUM 0
#define PWM_MIN 900
#define PWM_MAX 2100
#define PWM_VAL map(position_val,VAL_MIN,VAL_MAX,PWM_MIN,PWM_MAX)
```

The bottom, middle, and top of the Slide Switch are defined as MODE_0, MODE_1, and MODE_2 in each order.

Debouncing by variable counting

Sw_status : Recognizing if the switch is pressed
sw_cnt: When SW is On, recognizing after debouncing
cnt: When SW is off, recognizing after determining "off operation".

MODE_0 : Manual Position(Potentiometer)
MODE_1 : Position Select(A, B Position)
MODE_2 : Position Toggle(A,B Position)

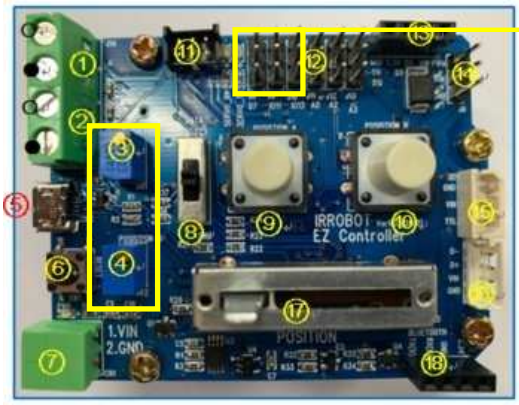
3.6. Example –Extra IO(1)

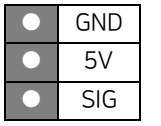
Controlling the actuator by receiving the input of the digital IO pin.
Select [Example] - [IRROBOT_EZController] - [EZ]–[EasyControl_ExtIO]

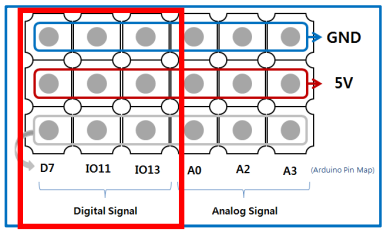
[Description]

- Mode switch (#8) can be positioned anywhere. (Mode switch only works in the built-in basic program)
- Supply power to the input power terminal (#7). Make sure correct input voltage(7.4V or 12V) and correct polarity.(GND and VCC)
- Carefully insert the connector suitable for the selected communication. (#14 PWM / #15 TTL / #16 RS-485)
  (For PWM connector (#14), refer to the page 6 to make sure correct polarity.)
- The left three of the upper header pins (# 12) are assigned for digital terminals, and the first & second pins from the left can be used.
- The signal terminal is the lowest pin of the 3 pins and the pin has a signal level of 5V. (See picture below)
  (Active Low / 5V : High Signal / 0V : Low Signal)
- Set the two points for position setting in the same way as the previous method through the two variable resistors on the left (#3 and 4).
- When a low signal is applied to the first pin, the actuator moves to the set point A,
  When a low signal is applied to the second pin, the actuator moves to the set B point.

[Program Description]







The left three pins are Arduino's Digital IO, the right three pins are Analog IO.

```
#include <IRROBOT_EZController.h>

#define ID_MAX 11
#define A_POSITION_VR Tester.VR_2
#define B_POSITION_VR Tester.VR_3
#define VR_MIN 0
#define VR_MAX 1023
#define VAL_MIN 0
#define VAL_MAX 4095
#define ID_NUM 0
#define PWM_MIN 900
#define PWM_MAX 2100
#define PWM_VAL map(position_val,VAL_MIN,VAL_MAX,PWM_MIN,PWM_MAX)


IRROBOT_EZController Tester(&Serial1);

short position_val;

void setup() {
  pinMode(7,INPUT);
  pinMode(11,INPUT);
  Tester.begin();
  Tester.Mightyzap.begin(32);
  Tester.setStep(ID_MAX,0,1023);
}
void loop() {
  unsigned char MightyZap_actID = ID_NUM;
  short A_stroke_val,B_stroke_val;

  A_stroke_val = map(A_POSITION_VR.read(),  VR_MIN, VR_MAX,  VAL_MIN, VAL_MAX);
  B_stroke_val = map(B_POSITION_VR.read(),  VR_MIN, VR_MAX,  VAL_MIN, VAL_MAX);

  if(digitalRead(7) == HIGH) position_val = A_stroke_val;
  else if(digitalRead(11) == HIGH) position_val = B_stroke_val;
  else position_val = position_val;
  Tester.Mightyzap.goalPosition(MightyZap_actID,position_val);
  Tester.servo_CH1.writeMicroseconds(PWM_VAL);
  delay(15);
}
```

Among Digital IO (7,11,13), 7 and 11 IO are declared as input.

When 5V signal is applied to pin 7, the actuator moves to the set point A. When 5V signal is applied to pin 11 the actuator moves to the set point B.

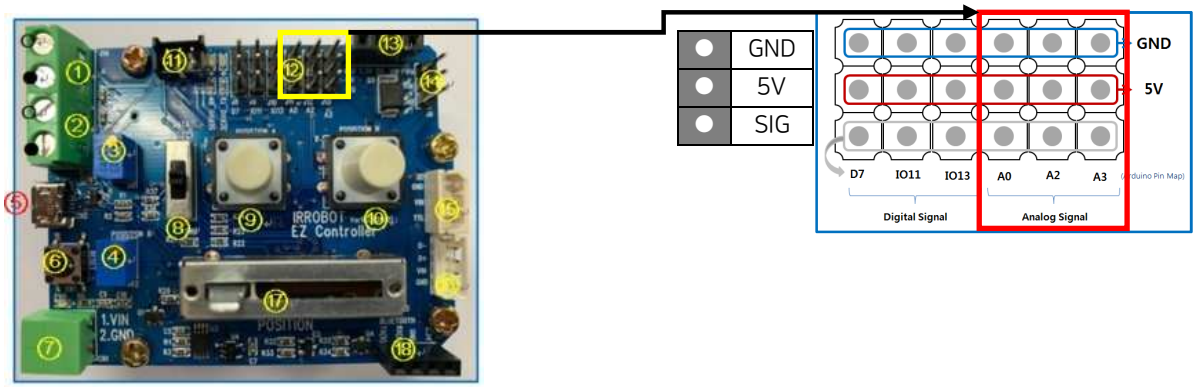
3.7. Example -Extra IO(2)

Controlling the actuator by receiving the input of an external sensor to the analog IO pin.
Select [Example] - [IRROBOT_EZController] - [EZ]-[EasyControl_Sensing]

[Description]

- Mode switch (#8) can be positioned anywhere. (Mode switch only works in the built-in basic program)
- Supply power to the input power terminal (#7). Make sure correct input voltage(7.4V or 12V) and correct polarity.(GND and VCC)
- Carefully insert the connector suitable for the selected communication. (#14 PWM / #15 TTL / #16 RS-485)
  (For PWM connector (#14), refer to the page 6 to make sure correct polarity.)
- Among the header pins (# 12), the right three are analog terminals, and these three pins are used for Extra IO. (See picture below)
- Connect the sensor to read the analog value to the pin according to the purpose.
- Actuator moves according to the sensor value.
  ex) CDS => brighter forward, darker backward

[Program Description]




```
#include <IRROBOT_EZController.h>

IRROBOT_EZController Controller(&Serial1);

#define ID_MAX 11
#define MANUAL_POSITION_VR Controller.VR_1
#define A_POSITION_VR Controller.VR_2
#define B_POSITION_VR Controller.VR_3
#define EXT_ANALOG_VR Controller.VR_4   //VR4 : A0  //VR5 : A2  //VR6 : A3
#define VR_MIN 0
#define VR_MAX 1023
#define VAL_MIN 0
#define VAL_MAX 4095
#define ID_NUM 0
#define PWM_MIN 900
#define PWM_MAX 2100
#define PWM_VAL map(position_val,VAL_MIN,VAL_MAX,PWM_MIN,PWM_MAX)

short position_val;

void setup(){
  Controller.begin();
  Controller.Mightyzap.begin(32);
  Controller.setStep(ID_MAX, 0, 1023);
}

void loop() {
  unsigned char MightyZap_actID = ID_NUM;
  short Ext_analog_val;

  Ext_analog_val = map(EXT_ANALOG_VR.read(), VR_MIN, VR_MAX, VAL_MIN, VAL_MAX);
  position_val = Ext_analog_val;
  Controller.Mightyzap.goalPosition(ID_NUM, position_val);
  Tester.servo_CH1.writeMicroseconds(PWM_VAL);
  delay(15);
}
```

In the library, analog pins A0, A2, A3 are defined as VR_4, VR_5, VR_6.
Declare the pin you want to use with a defined name, or directly declare matched analog pin.

3.8. Example –External Communication

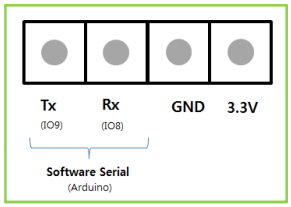
Controlling the actuator through external communication by the Bluetooth port.
Select [Example] - [IRROBOT_EZController] - [EZ]-[EasyControl_ExtCom]

[Description]

- Mode switch (#8) can be positioned anywhere. (Mode switch only works in the built-in basic program)
- Supply power to the input power terminal (#7). Make sure correct input voltage(7.4V or 12V) and correct polarity.(GND and VCC)
- Carefully insert the connector suitable for the selected communication. (#14 PWM / #15 TTL / #16 RS-485)
- (For PWM connector (#14), refer to the page 6 to make sure correct polarity.)
- There is a port (# 18) for external communication at the bottom right, and connects the device to the pin according to the pin arrayal.
- If you press the A(B) button in the example, 'A(B)' is sent as an ASCII value.
- When the board receives 'A' as an ASCII value, the actuator moves to the A point set the same as before, Similarly, when the ASCII value 'B' is received, the actuator moves to the set point B.

[Program Description]

Since it is based on Bluetooth, the communication voltage level is 3.3V.




```
#include <IRROBOT_EZController.h>

IRROBOT_EZController Controller(&Serial1);
SoftwareSerial userSerial(8,9);

#define ID_MAX 11
#define MANUAL_POSITION_VR Controller.VR_1
#define A_POSITION_VR Controller.VR_2
#define B_POSITION_VR Controller.VR_3
#define EXT_ANALOG_VR Controller.VR_4
#define VR_MIN 0
#define VR_MAX 1023
#define VAL_MIN 0
#define VAL_MAX 4095
#define IS_A_POSITION_ON Controller.POS_A.isOFF()
#define IS_B_POSITION_ON Controller.POS_B.isOFF()
#define ID_NUM 0
#define PWM_MIN 900
#define PWM_MAX 2100
#define PWM_VAL map(position_val,VAL_MIN,VAL_MAX,PWM_MIN,PWM_MAX)

short position_val;
char RxChar;
short A_stroke_val,B_stroke_val;

void setup() {
  Controller.begin();
  Controller.Mightyzap.begin(32);
  Controller.setStep(ID_MAX,0,1023);

  Serial.begin(9600);
  userSerial.begin(9600);
}

void loop() {
  unsigned char MightyZap_actID = ID_NUM;

  A_stroke_val = map(A_POSITION_VR.read(),  VR_MIN, VR_MAX,  VAL_MIN, VAL_MAX);
  B_stroke_val = map(B_POSITION_VR.read(),  VR_MIN, VR_MAX,  VAL_MIN, VAL_MAX);
  Sw_Func();
  ExtComData_Listen();
  ExtComData_Func();
}
```

```
void Sw_Func(void){
  if(IS_A_POSITION_ON){
    userSerial.write('A');
    Serial.println("'A' Send");
    delay(500);
  }
  else if(IS_B_POSITION_ON){
    userSerial.write('B');
    Serial.println("'B' Send");
    delay(500);
  }
}

void ExtComData_Listen(void){
  if(userSerial.available()>0){
    RxChar = userSerial.read();
  }
}

bool ExtComData_Func(){
  if(RxChar == 'A'){
    position_val = A_stroke_val;
    Serial.println("'A' Recieved");
  }
  else if(RxChar == 'B'){
    position_val = B_stroke_val;
    Serial.println("'B' Recieved");
  }
  Controller.Mightyzap.goalPosition(ID_NUM, position_val);
  Controller.servo_CH1.writeMicroseconds(PWM_VAL);
}
```

On the Arduino, IO8 and IO9 pins are declared as RX and TX respectively.

Communication data is received byte by byte through "ExtComData_Listen ()" function.

Set baud rate of serial monitor with Serial.begin () function
Check the data which is sent and received by the Sw_func () function and ExtComData_Func () function on the serial monitor.

The Sw_Func () function sends ASCII code values corresponding to A and B respectively, and the ExtComData_Func () function moves the actuator to the position corresponding to the received ASCII code value.

3.9. Example – Mode Selection

Example for the mode setting that can be operated according to the value of the mode selection switch (Fig# 8).
Select [Example] - [IRROBOT_EZController] - [EZ]-[EasyControl _ModeSelect]

[Description]

In the same way as [Basic Function], the operation can be different according to the position of Mode selection switch(#8), and the operation can be assigned for each mode.

[Program Description]


```
#include <IRROBOT_EZController.h>

#define ID_MAX 11

#define ModeSW_1 Tester.MODE_0.isOFF()
#define ModeSW_2 Tester.MODE_1.isOFF()
#define ModeSW_3 Tester.MODE_2.isOFF()

#define MANUAL_MODE 1
#define POS2_MODE 2
#define TOGGLE_MODE 3
#define EXT_IO_MODE 4
#define EXT_SENSING_MODE 5
#define EXT_COM_MODE 6

IRROBOT_EZController Tester(&Serial1);


void setup() {
  Tester.begin();
  Tester.Mightyzap.begin(32);
  Tester.setStep(ID_MAX,0,1023);
}
void loop()
{
  int sw_val;
  if(ModeSW_1)sw_val = 1;
  else if(ModeSW_2) sw_val = 2;
  else if(ModeSW_3) sw_val = 3;
  Tester.ModeSelect(MANUAL_MODE,POS2_MODE,TOGGLE_MODE,sw_val);
  delay(10);
}
```

Index values for each mode are defined sequentially.

1 : when the mode switch is at the bottom
2 : mode switch at the middle / 3 : at the top

Tester.ModeSelect(int mode1, int mode2. int mode3, int sw);
mode1 : operation when sw value is 1
mode 2 : operation when sw value is 2
mode3 : operation when sw value is 3
sw : Mode value set by mode switch

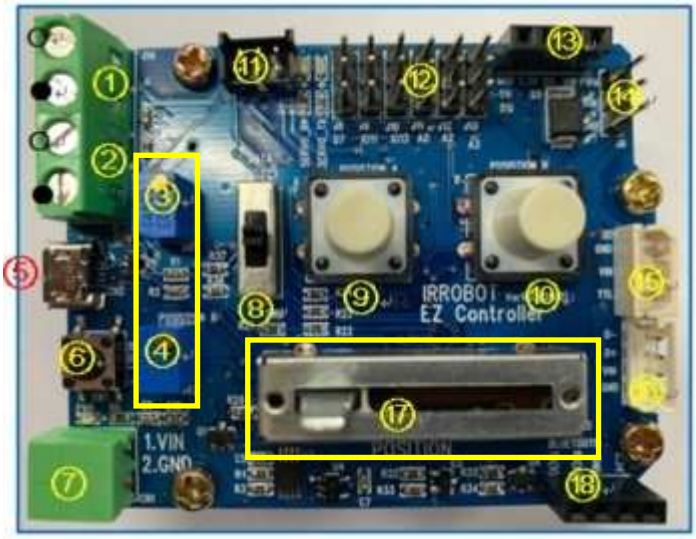
3.10. Example –Stroke Limit

Example which can set the max/min limit of stroke with 2 x position setting variable resistors(fig#3&4).
Select [Example] - [IRROBOT_EZController] - [EZ]–[EasyControl _StrokeLimit]

[Description]

- Mode switch (#8) can be positioned anywhere. (Mode switch only works in the built-in basic program)
- Supply power to the input power terminal (#7). Make sure correct input voltage(7.4V or 12V) and correct polarity.(GND and VCC)
- Carefully insert the connector suitable for the selected communication. (#14 PWM / #15 TTL / #16 RS-485)
- (For PWM connector (#14), refer to the page 6 to make sure correct polarity.)
- Set the position limit by turning the variable resistors (Fig# 3 and 4) that adjust the position of the max and min points of the stroke in the clockwise (-) and counterclockwise (+) directions. Clockwise direction is for retraction (short stroke), Counterclockwise direction is for extension(long stroke).
- Operate actuator by adjusting linear potentiometer (Fig#17).
  Operating position will be affected by position setting V/R(#3&4), and even if the potentiometer is moved, the position is controlled only within the section set by the variable resistor.

[Program Description]




```
#include <IRROBOT_EZController.h>

#define ID_MAX 11
#define MANUAL_POSITION_VR Tester.VR_1
#define A_POSITION_VR Tester.VR_2
#define B_POSITION_VR Tester.VR_3
#define IS_A_POSITION_ON Tester.POS_A.isOFF()
#define IS_B_POSITION_ON Tester.POS_B.isOFF()
#define SW_A Tester.POS_A
#define SW_B Tester.POS_B

#define VR_MIN 0
#define VR_MAX 1023
#define VAL_MIN 0
#define VAL_MAX 4095
#define ID_NUM 0
#define PWM_MIN 900
#define PWM_MAX 2100
#define PWM_VAL map(position_val,VAL_MIN,VAL_MAX,PWM_MIN,PWM_MAX)

IRROBOT_EZController Tester(&Serial1);

short position_val;
bool tg_flag,Sw_status = 1;
int sw_cnt = 0;

void setup() {
  Tester.begin();
  Tester.Mightyzap.begin(32);
  Tester.setStep(ID_MAX,0,1023);
}
```

```
void loop()
{
  unsigned char MightyZap_actID = ID_NUM;
  short Manual_position_val,A_stroke_val,B_stroke_val;
  int A_stroke_limit, B_stroke_limit, stroke_limit_dir;
  int short_stroke_limit,long_stroke_limit;

  Manual_position_val  = map(MANUAL_POSITION_VR.read(),VR_MIN,VR_MAX,VAL_MIN,VAL_MAX);
  short_stroke_limit = map(A_POSITION_VR.read(),  VR_MIN, VR_MAX,  VAL_MIN, VAL_MAX);
  long_stroke_limit = map(B_POSITION_VR.read(),  VR_MIN, VR_MAX,  VAL_MIN, VAL_MAX);

  if(short_stroke_limit>long_stroke_limit)
  {
    int temp = short_stroke_limit;
    short_stroke_limit = long_stroke_limit;
    long_stroke_limit = temp;
  }

  if(Manual_position_val<short_stroke_limit) Manual_position_val = short_stroke_limit;
  else if(Manual_position_val>long_stroke_limit) Manual_position_val = long_stroke_limit;
  position_val = Manual_position_val;

  Tester.Mightyzap.goalPosition(MightyZap_actID,position_val);
  Tester.servo_CH1.writeMicroseconds(PWM_VAL);
  delay(10);
}
```

Regardless of V/R(#3&4) location, large value sets max limit and small value sets min limit.

Determine the potentiometer value so that linear potentiometer does not exceed the set range.