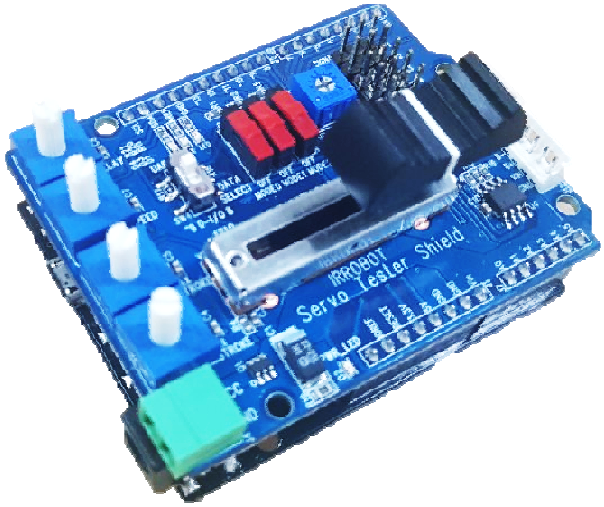
Robust Mini Linear Servo Motor ——

mightyZAP Servo Tester Shield User Manual



ARDUINO

IR ROBOT
Imagination to Realization

INDEX

1 Outline

1.1 Introduction

IR-STS01, Servo Tester Shield is Arduino Shield to operate/test mightyZAP more easily on Arduino Uno & Leonardo. User may connect/operate/test mightyZAP linear servo motor with IR-STS01 without PC connection.

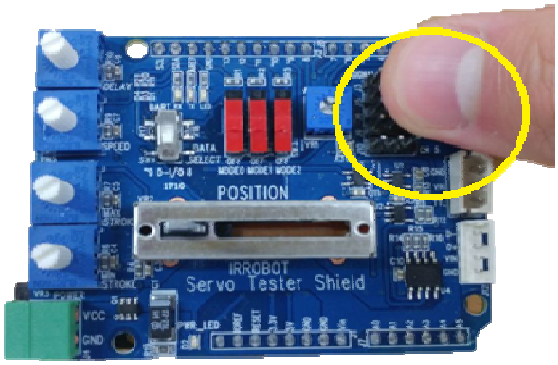
Main features:

- Servo Test : User may test mightyZAP linear servo motions using basic motion program stored in the tester.
- User motion : User may create / control user's motion using our Servo Arduino Library.
- Function as a main board : No need to make additional control circuitry for data communication. Just plug & operate using various I/O ports.

1.2 Caution

Peruse below matters to protect the damage of products and secure warranty service.

1. Please do not supply power to IR-STS01 when it is placed on the un-insulated material such as a metal plate. It causes serious damage on the product due to the short-circuit.
2. Be careful when you assemble the board and connect the wires. Do not put excessive force to the component.
3. Make sure input voltage of your servo motor as well as polarity of power connection. For example, if your servo input voltage is 7.4V, you need to apply 7.4V electricity and 12V needs to be applied for 12V input servo motor.
4. When servo motor is connected with IR-STS01, make sure pin polarity. Otherwise, servo motor can be damaged.
5. Use proper standard connector and pins to protect mis-connection which lead servo motor damage.



6. Avoid product from fall-down, fire, hot stuff, water, dust or oil.
7. Indoor use only.
8. Keep away from chidren or animals.

1.3 Storage

Avoid from below environment and store properly.

- Hot temperature more than 60 ℃ or extremely low temperature below minus 20℃.
- Direct light or Fire
- High humidity and dust
- Vibration
- Static electricity

2 Servo Motor Connection

2.1. Assembly

IR-STS01 is an Arduino shield designed for Arduino Uno and Arduino Leonardo. Assemble Test Shield onto the Arduino Board gently aligning pin structure.

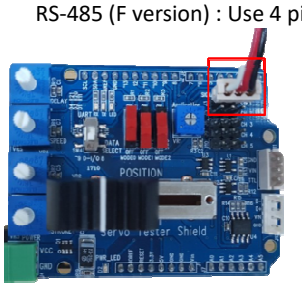
2.2. Linear Servo Connection

There are 3 different communication method with IR-STS01.

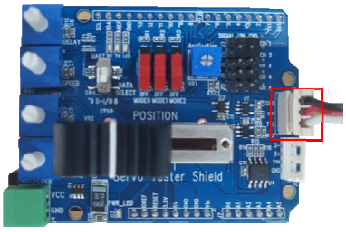
Check out your servo's communication mode and connect to proper port accordingly.

[(Ex) PT version like L12-20**PT**-3 : PWM/TTL ,F like L12-20**F**-3 Version : RS-485]

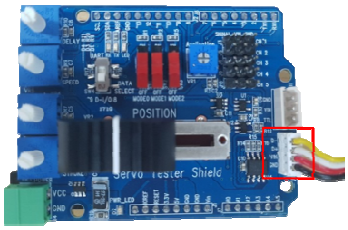
1. PWM(PT version) : Use 3 pin wire connector and connect the servo to CH1 ~ CH5 ports as below.
2. TTL (PT version) : Use 3 pin wire connector and connect the servo to TTL port as below.
3. RS-485 (F version) : Use 4 pin wire connector and connect the servo to RS-485 port as below.



< 3Pin PWM Port>

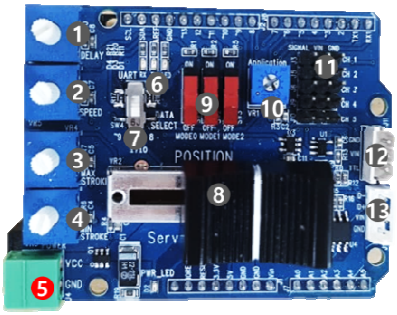


< 3Pin TTL Communication Port>



< 4pin RS-485 Communication Port>

2.3. Power Connection



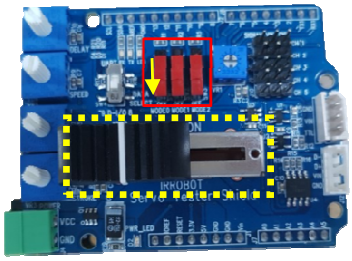
To operate servo motor with IR-STS01, power needs to be supplied by proper method like power supply, power adaptor, or battery.

Connect power to Power input port(#5 on the photo). Thanks to protection circuitry, servo motor can be protected when wrong polarity is applied, but servo motor will not be operated.

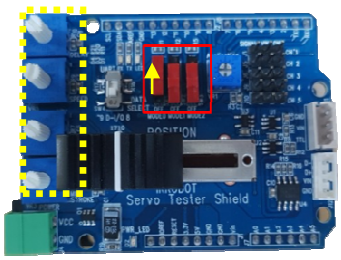
Make sure input voltage of your servo motor as well as polarity of power connection. For example, if your servo input voltage is 7.4V, you need to apply 7.4V electricity and 12V needs to be applied for 12V input servo motor.

3 Basic Test Operation

- Test the motion of mightyZAP servo motor using basic operation program stored in IR-STS01.
- There are two modes in basic operation program. - Auto Mode and Manual Mode.

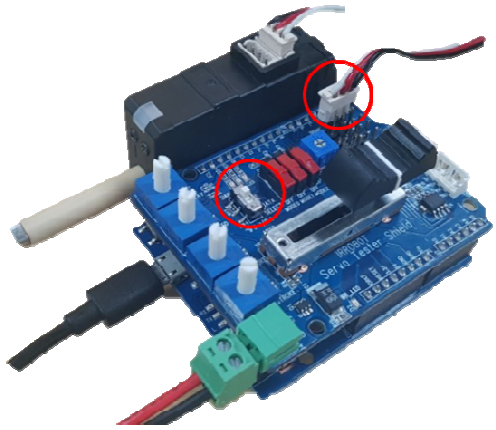


<Manual mode>



<Auto mode>

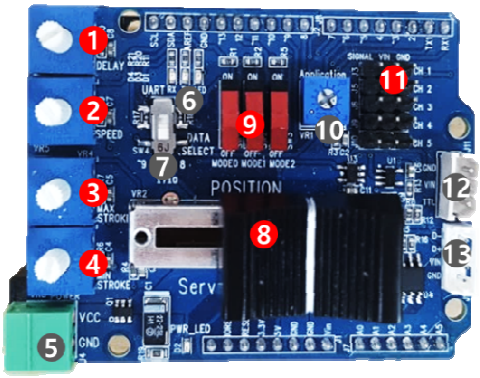
- Manual mode : Connect the power and Connect the servo motor with IR-STS01. Slide DOWN “Mode 0 Switch(Red marked above)” to set “Maunal mode”. Then, move black lever(yellow marked above) side to side to operate the servo motor.
- Auto mode : Connect the power and Connect the servo motor with IR-STS01. Slide UP “Mode 0 Switch(Red marked above)” to set “AUTO mode”. Using white rotary volumes on left side and red slide switches, user is able to control position, speed and max/min stroke limit.



- Please make sure that Data select switch must be located at UART side all the time. (see the photo left)

3.1. Manual Mode (PWM Control)Operation

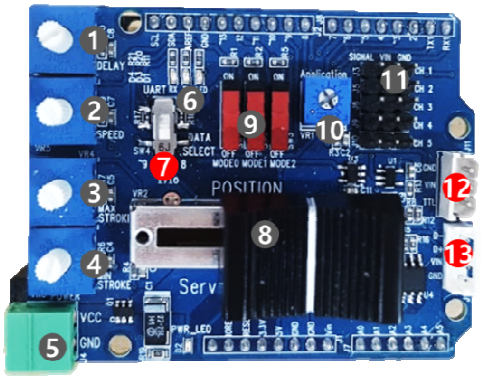
Slide down “Mode 0” switch. Connect servo motor to Ch.1 using 3pin connector and control position using Position volume (#8 below).



- Rotary volume#1 : Under manual mode, this white rotary volume#1 is to control position for the servo motor which is connected to CH.2. Under manual mode, this volume is nothing to do with delay function.
- Rotary volume#2 : Under manual mode, this white rotary volume#2 is to control position for the servo motor which is connected to CH.3. Under manual mode, this volume is nothing to do with Speed function.
- Rotary volume#3 : Under manual mode, this white rotary volume#3 is to control position for the servo motor which is connected to CH.4. Also, user may set Max. stroke limit of the servo motor at CH.1.
- Rotary volume#4 : Under manual mode, this white rotary volume#4 is to control position for the servo motor which is connected to CH.5. Also, user may set Min. stroke limit of the servo motor at CH.1.
  (In case that Min Stroke value is bigger than Max Stroke value on CH.1 servo motor, IR-STS01 will substitute each value.)
- Slide volume#8 : As stated above, Control position which is connected to CH.1.
- Mode Switches#9 : If Mode 1 switch slides UP, Min/Max Stroke setting value of Ch.1 will Not be applied.
- 3 pin ports#11 : PWM ports for each Channel(Ch1~5)

3.2. Manual Mode (RS-485/TTL Control) Operation

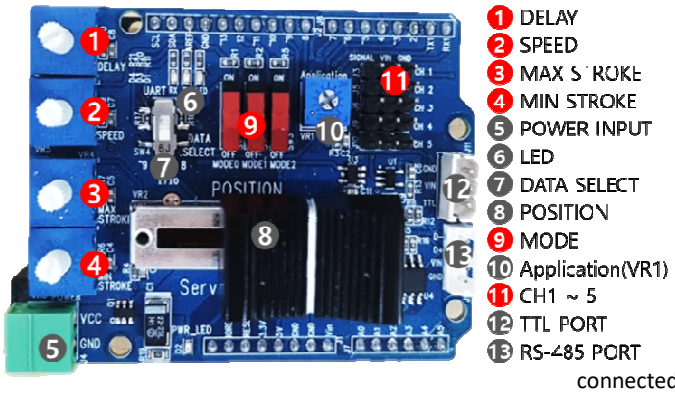
Slide down “Mode 0” switch. Connect servo motor to TTL(3pins) or RS-485(4pins) port and control position using Position volume (#8 below).



- TTL PORT#12 : Connect PT version servo motor using 3 pin connector wire.
- RS-485 PORT#13 : Connect F version servo motor using 4pin connector wire.
- DATA SELECT#7 : For TTL/RS-485 communication, this swtich must be at UART side all the time. When this swtich slides down, 3 pin and 4 pin connector ports will be used for I/O devices such as sensors, etc. So, make sure this switch slides UP all the time for TTL/RS-485 communication.
- Rotary volume#3 : User may set Max. stroke limit of the servo motor connected to TTL or RS-485 ports.
- Rotary volume#4 : User may set Min. stroke limit of the servo motor connected to TTL or RS-485 ports.
- Slide volume#8 : As stated above, Control position which is connected to TTL or RS-485 ports.
- Mode Switches#9 : If Mode 1 switch slides UP, Min/Max Stroke setting value of connected servo motor will Not be applied.

3.3. Auto Mode(PWM/RS-485/TTL Control) Operation

Under Auto mode, user may create their own motion in limited range and operate it automatically.
Slide UP Mode 0 switch to set Auto mode. Connect servo motors to PWM or TTL or RS-485 ports accordingly.



- Rotary volume#1 (Delay) : Adjust “Delay” value between the motion (between the commands) for CH.1 servo motor(PWM) or servo motor connected to TTL/RS-485 port. Also, this volume can be used to control poition for servo motor connected to Ch.2 PWM port.
- Rotary volume#2 (Speed) : Adjust “Speed” value for CH.1 servo motor(PWM) or servo motor connected to TTL/RS-485 port. Also, this volume can be used to control poition for servo motor connected to Ch.3 PWM port.
- Rotary volume#3 (MAX STROKE) : Adjust “Max stroke limit” of CH.1 servo motor(PWM) or servo motor connected to TTL/RS-485 port. Also, this volume can be used to control poition for servo motor connected to Ch.4 PWM port.
- Rotary volume#4 (MIN STROKE) : Adjust “Min stroke limit” of CH.1 servo motor(PWM) or servo motor connected to TTL/RS-485 port. Also, this volume can be used to control poition for servo motor connected to Ch.5 PWM port.
  (In case that Min Stroke value is bigger than Max Stroke value on CH.1 servo motor, IR-STS01 will substitute each value.)
- LED#6 : LED will be ON in case of Auto Mode.
- Mode Switches#9 : If Mode 1 switch slides UP, Min/Max Stroke setting value of connected servo motor will Not be applied.
- Application(VR1)#10 : Asign Servo ID from 0~9. (n=ID#n. In clock-wise, 0-9)
- 3 pin ports#11 : PWM ports for each Channel(Ch1~5)
- 4 pin ports#12 and #13 : TTL(3pins - #12) and RS-485(4pins - #13) ports.

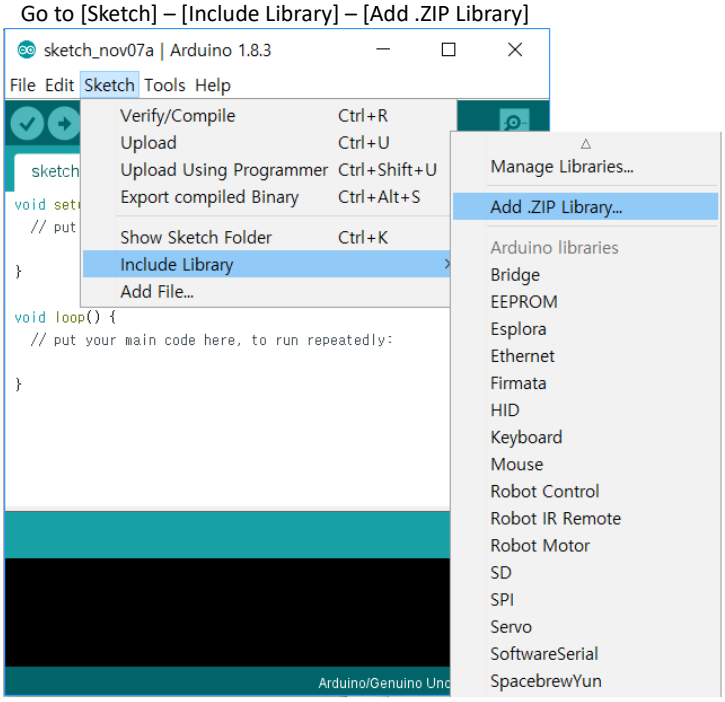
4 Use Arduino Library

Our dedicated Arduino API can be downloaded from our website at http://www.irrobot.com/ (➔ Go to "Digital achieves"➔ Linear servo motors). Basic motion is already stored in IR-STS01 and user is able to test servo motors with basic motion. Arduino API library needs to be downloaded if users want to create their own motion using IR-STS01 on Arduino IDE.
To use Arduino API, follow below steps.

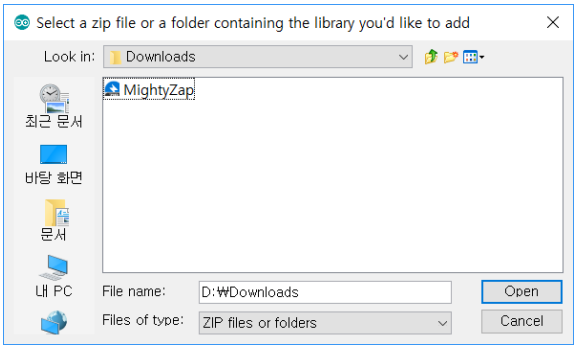
1. Add Arduino IDE MightyZap Library
2. Load MightyZap Example
3. Program Upload

4.1 Library Adding

1. Download "MightyZap.Zip" and "IRROBOT_ServoTesterShield.Zip" from our website.
2. Go to [Sketch] – [Include Library] – [Add .ZIP Library]



3. Select "MightyZap.Zip"



4. Add "IRROBOT_TESTER_SHIELD.Zip" in same way as above.

4.2 Example Load

1. Start Arduino IDE
2. [File] - [Example] - [IRROBOT_ServoTesterShield] – Select desired example.

4.3 Program Upload

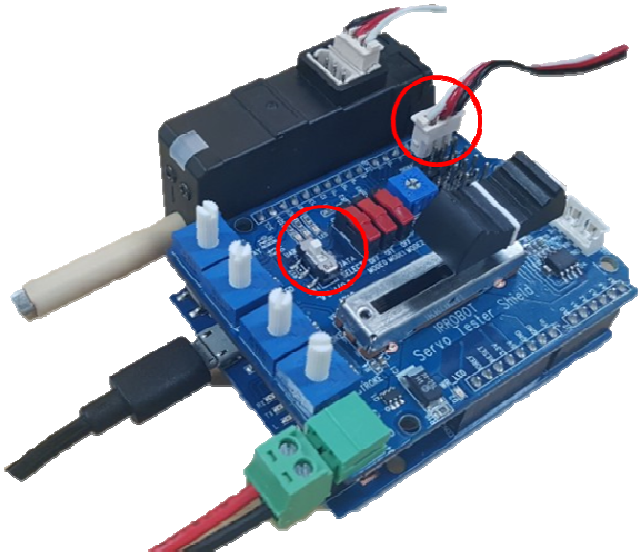
Upload proper Examples according Arduino Board brand. (UNO or Leonardo)

For UNO Board :
[File] - [Example] - [IRROBOT_ServoTesterShield] – [UNO]- [ServoTester_HardwareSerial]
For Leonardo Board :
[File] - [Example] - [IRROBOT_ServoTesterShield] – [LEO]- [ServoTester_HardwareSerial]

5 Shield Control Example via Arduino IDE

Control I/O ports of IR-STS01 using Arduino IDE. Users may assign their own function for each I/O port(switches, rotary volumes, etc) on the IR-STS01.

5.1 Outline / Caution

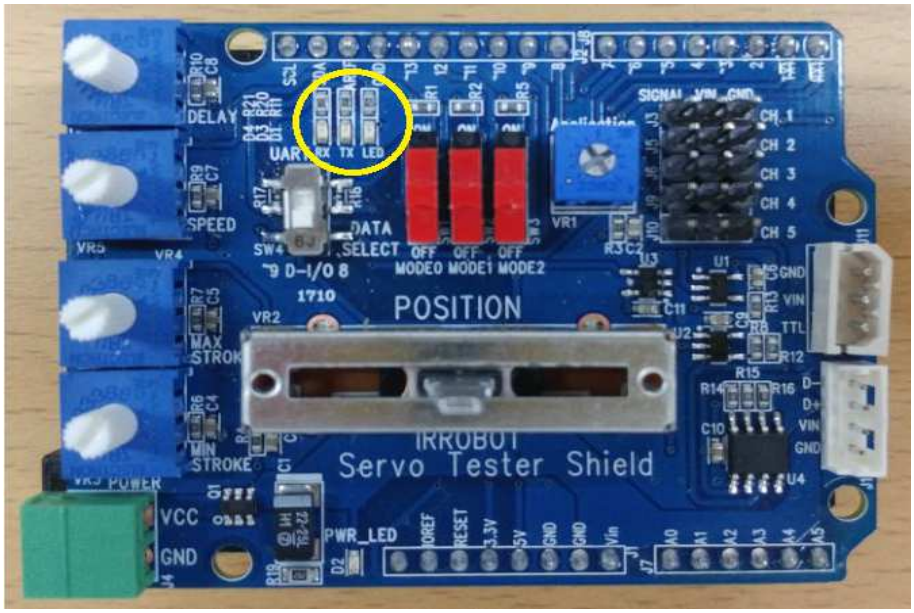


Connect mightyZAP to Ch.1 of PWM port or TTL or RS-485 port.
To upload example, make sure if “data select” switch is slide UP.

Please note, once users assign their own function, basic program stored at the factory will be removed and user program will be re-wrote. If user needs Basic Program again, user may load it from the library we provide.

5.2. Example - User LED

Control LED on IR-STS01.



Example for periodical flickering of LED

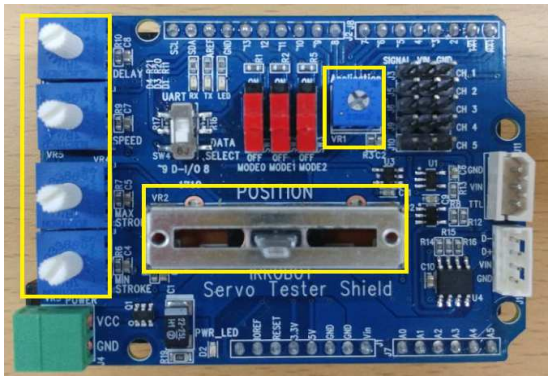
```
#include <IRROBOT_ServoTesterShield.h>

SoftwareSerial mySerial(8,9);
IRROBOT_ServoTesterShield Tester(&mySerial);

void setup() {
  Tester.begin();
}

void loop() {
  Tester.onLED();
  delay(1000);
  Tester.offLED();
  delay(1000);
}
```

Source to flicker LED periodically.

Control LED using “onLED(), offLED” function.

5.3.Example- VR Read

There are 6 x VRs on IR-STS01. User may control stroke, delay, speed and position of mightyZAP using these VRs.



Example for VR value output on Serial monitor

```
#include <IRROBOT_ServoTesterShield.h>

SoftwareSerial mySerial(8,9);
IRROBOT_ServoTesterShield Tester(&mySerial);

#define APPLICATION_VR      Tester.VR_1
#define MANUAL_POSITION_VR  Tester.VR_2
#define MIN_STROKE_VR       Tester.VR_3
#define MAX_STROKE_VR       Tester.VR_4
#define SPEED_VR            Tester.VR_5
#define DELAY_VR            Tester.VR_6

void setup() {
  Serial.begin(9600);
  Tester.begin();
}

void loop() {
  Serial.print("APPLICATION = ");
  Serial.println(APPLICATION_VR.read());
  Serial.print("MANUAL POSITION = ");
  Serial.println(MANUAL_POSITION_VR.read());
  Serial.print("MIN STROKE= ");
  Serial.println(MIN_STROKE_VR.read());
  Serial.print("MAX STROK = ");
  Serial.println(MAX_STROKE_VR.read());
  Serial.print("SPEED = ");
  Serial.println(SPEED_VR.read());
  Serial.print("DELAY = ");
  Serial.println(DELAY_VR.read());
  Serial.println();
  delay(1000);
}
```

Source to display VR (VR1~VR6) value on Serial monitor when user changes VR physically.

Buad rate setting of serial monitor with “Serial.begin()” function.

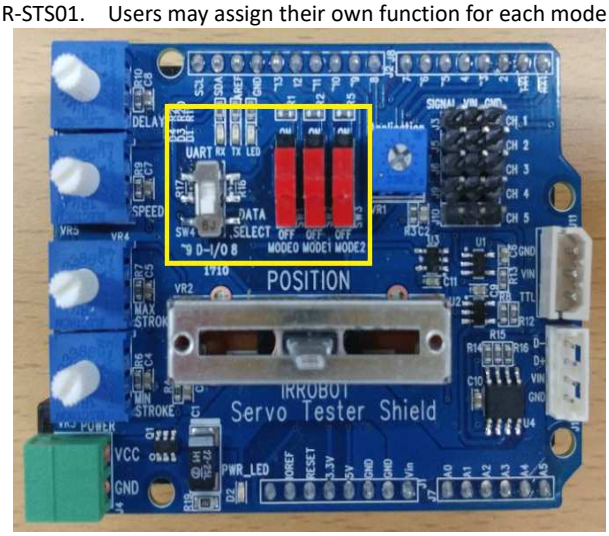
Read ADC value with “read()” function.
Output ADC value with “Serial.print()” function.

Range of VR value output is between 0~1023.

To open Serial monitor, select [Tool]-[Serial Monitor].

5.4. Example- SWITCH

There are 3 x red slide switches on IR-STS01. Users may assign their own function for each mode switch to make various functions.



Example to output logical value of mode switches on Serial monitor

```
#include <IRROBOT_ServoTesterShield.h>

SoftwareSerial mySerial(8,9);
IRROBOT_ServoTesterShield Tester(&mySerial);

void setup() {
  Serial.begin(9600);
  Tester.begin();
}

void loop() {
  Serial.print("MODE0 = ");
  Serial.println(Tester.MODE_0.read());
  Serial.print("MODE1 = ");
  Serial.println(Tester.MODE_1.read());
  Serial.print("MODE2 = ");
  Serial.println(Tester.MODE_2.read());
  Serial.println();
  delay(1000);
}
```

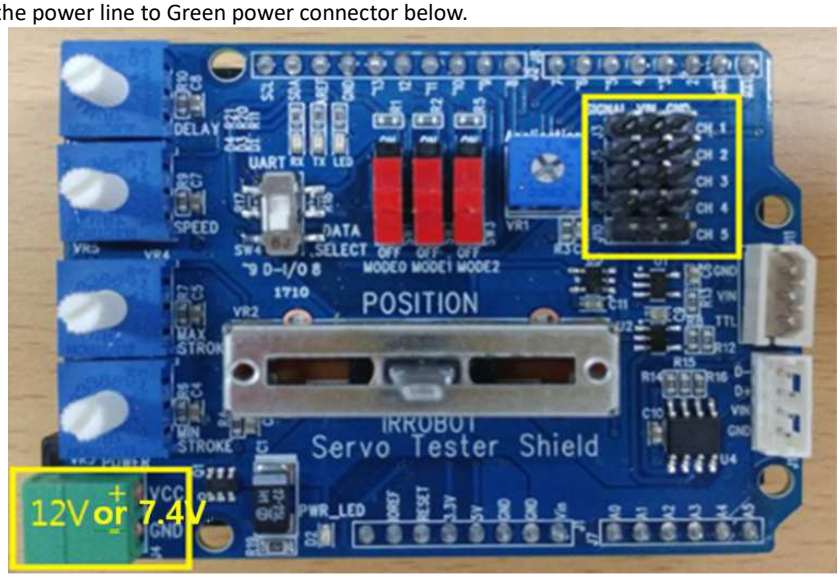
Example to output logical value of mode switch on serial monitor.

-Buad rate setting of serial monitor with “Serial.begin()” function.
-Read logical value with “read()” function.
-Output logical value with “print()” function.

User is able to monitor change of logical value on Serial monitor when each switch changes its position.

5.5.Example-Servo Motor PWM Control 1

There are 5 x PWM channels on IR-STS01 to control “PT” version mightyZAP. To operate mightyZAP Servo motor, user needs to connect proper power (12V or 7.4V according to the input voltage of user’s servo). Power source can be Power supply, power adaptor, or battery and connect the power line to Green power connector below.



Example to control mightyZAP using PWM

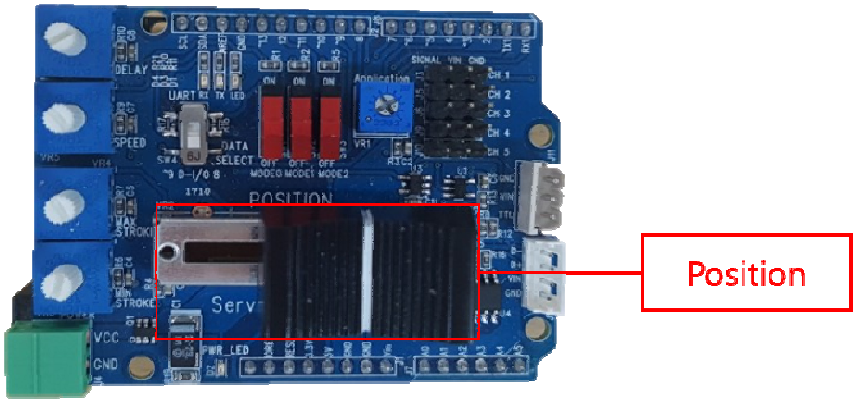
```
#include <IRROBOT_ServoTesterShield.h>

IRROBOT_ServoTesterShield Tester(&Serial);

int Position;

void setup() {
  Tester.begin();
}

void loop() {
  for(Position = 0 ; Position <= 180 ; Position += 1)
  {
    Tester.servo_CH1.write(Position);
    delay(15);
  }
  for(Position = 180 ; Position >= 0 ; Position -= 1)
  {
    Tester.servo_CH1.write(Position);
    delay(15);
  }
}
```

Prepare test by opening new file on Arduino IDE.

Control position of servo motor using “Servo_CH1.write()” function. Position range is between 0~180.

5.6.Example-Servo Motor PWM Control 2

Select [IRROBOT_ServoTesterShield] – [UNO] or[LEO] –[ServoTester_ Knob]
Read VR value using “analogRead()” command to control servo motor position.



Example to control servo motor using VR

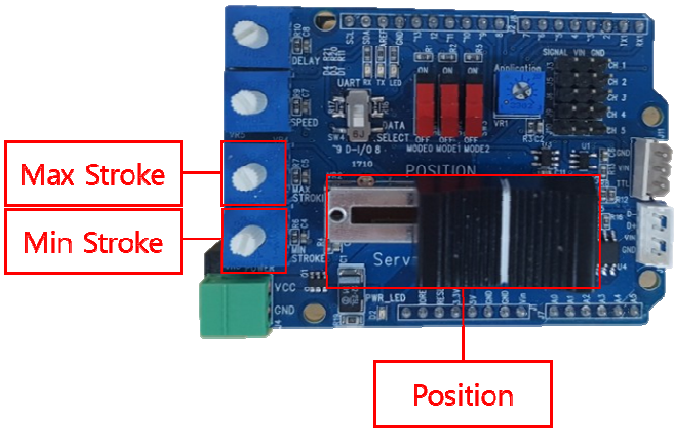
```
#include <IRROBOT_ServoTesterShield.h>
#define MANUAL_POSITION_VR  myservo.VR_2

IRROBOT_ServoTesterShield Tester(&Serial);

int Manual_positon_val;

void setup() {
  Tester.begin();
}

void loop() {
  Manual_positon_val = MANUAL_POSITION_VR.read();
  Manual_positon_val = map(Manual_positon_val, 0, 1023, 0, 180);
  Tester.servo_CH1.write(Manual_positon_val);
  delay(15);
}
```

Source to control servo motor using VR value.

Read VR value by “read()” function to control servo motor position.
Read value range is between 0~1023.
Adjust scale value between 0~180 to control servo motor

5.7.Example - Stroke Limit (PWM Control)

Select [IRROBOT_ServoTesterShield] – [UNO] or [LEO] –[ServoTester_StrokeLimit]
Control Max / Min stroke limit by Rotary volumes#3 and 4.




```
#include <IRROBOT_ServoTesterShield.h>

SoftwareSerial mySerial(8,9);
IRROBOT_ServoTesterShield Tester(&mySerial);

#define APPLICATION_VR      Tester.VR_1
#define MANUAL_POSITION_VR  Tester.VR_2
#define MIN_STROKE_VR       Tester.VR_3
#define MAX_STROKE_VR       Tester.VR_4
#define SPEED_VR            Tester.VR_5
#define DELAY_VR            Tester.VR_6

int Manual_positon_val;
int MIN_STROKE_VAL;
int MAX_STROKE_VAL ;

void setup() {
  Serial.begin(9600);
  Tester.begin();
}

void loop() {
  //Reads analoge
  MIN_STROKE_VAL = MIN_STROKE_VR.read();
  MAX_STROKE_VAL = MAX_STROKE_VR.read();
  Manual_positon_val =MANUAL_POSITION_VR.read();

  // position Limit
  if(Manual_positon_val>=MAX_STROKE_VAL)
    Manual_positon_val = MAX_STROKE_VAL;
  else if(Manual_positon_val<=MIN_STROKE_VAL)
    Manual_positon_val = MIN_STROKE_VAL;

  Manual_positon_val = map(Manual_positon_val, 0, 1023, 0, 180);

  Tester.servo_CH1.write(Manual_positon_val);
  delay(15);
}
```

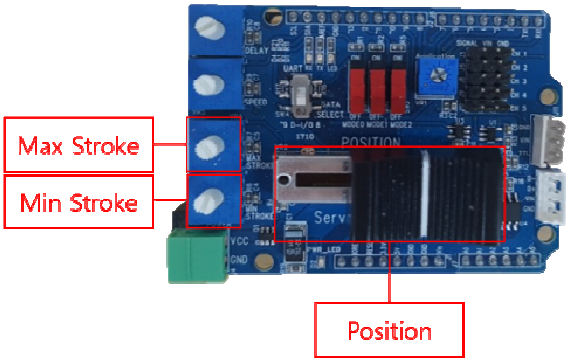
Source to set Max/Min stroke limit by reading two VR values.

Read each VR value by “Read()” function to set Max/Min stroke limit. Read value range is between 0~1023.

Compare stroke limit value with goal position value. If goal position value is out of stroke limit, goal position will be replaced with stroke limit.

5.8. VR Reverse (PWM Control)

Select [IRROBOT_ServoTesterShield] – [UNO/LEO] –[ServoTester_VR_Reverse]
Control Max / Min stroke limit by Rotary volumes#3 and 4.



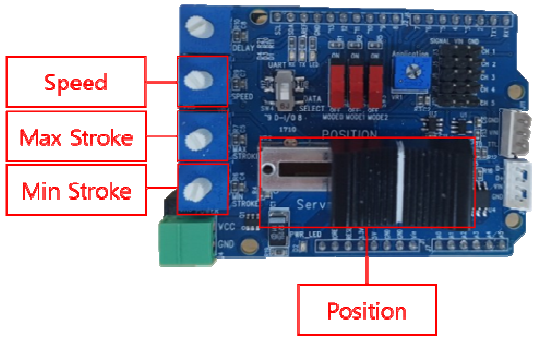

```
#include <IRROBOT_ServoTesterShield.h>

SoftwareSerial mySerial(8,9);
IRROBOT_ServoTesterShield Tester(&mySerial);

#define APPLICATION_VR      Tester.VR_1
#define MANUAL_POSITION_VR  Tester.VR_2
#define MIN_STROKE_VR       Tester.VR_3
#define MAX_STROKE_VR       Tester.VR_4
#define SPEED_VR            Tester.VR_5
#define DELAY_VR            Tester.VR_6

int position_val;
int MIN_STROKE_VAL, min_stroke_limit;
int MAX_STROKE_VAL, max_stroke_limit ;

void setup() {
  Serial.begin(9600);
  Tester.begin();
}

void loop() {
  //Reads analoge
  MIN_STROKE_VAL = MIN_STROKE_VR.read();
  MAX_STROKE_VAL = MAX_STROKE_VR.read();
  position_val   = MANUAL_POSITION_VR.read();

  //VR Resverse
  if(MAX_STROKE_VAL<MIN_STROKE_VAL) {
    min_stroke_limit=MAX_STROKE_VAL;
    max_stroke_limit=MIN_STROKE_VAL;
  }else {
    min_stroke_limit=MIN_STROKE_VAL;
    max_stroke_limit=MAX_STROKE_VAL;
  }

  // position Limit
  if(position_val>=max_stroke_limit)
    position_val = max_stroke_limit;
  else if(position_val<=min_stroke_limit)
    position_val = min_stroke_limit;

  position_val = map(position_val, 0, 1023, 0, 180);
  Tester.servo_CH1.write(position_val);
  delay(15);
}
```

Source to set Max/Min stroke limit by reading two VR values.

Compare max and min stroke limit and replace its value if min value is bigger than max value.

5.9. Example - VR Speed (PWMControl)

Select [IRROBOT_ServoTesterShield] – [UNO]or[LEO] –[ServoTester_VR_Speed]
Control Speed of servo motor by rotary knob#2.




```
#include <IRROBOT_ServoTesterShield.h>

SoftwareSerial mySerial(8,9);
IRROBOT_ServoTesterShield Tester(&mySerial);

#define APPLICATION_VR      Tester.VR_1
#define MANUAL_POSITION_VR  Tester.VR_2
#define MIN_STROKE_VR       Tester.VR_3
#define MAX_STROKE_VR       Tester.VR_4
#define SPEED_VR            Tester.VR_5
#define DELAY_VR            Tester.VR_6

int goal_position,pre_position;
int position_val=0;
int MIN_STROKE_VAL, min_stroke_limit;
int MAX_STROKE_VAL, max_stroke_limit ;
int SPEED_VAL;

void setup() {
  Serial.begin(9600);
  Tester.begin();
}

void loop() {
  //Reads analoge
  MIN_STROKE_VAL = MIN_STROKE_VR.read();
  MAX_STROKE_VAL = MAX_STROKE_VR.read();
  goal_position  = MANUAL_POSITION_VR.read();
  SPEED_VAL      = SPEED_VR.read();

  SPEED_VAL = map(SPEED_VAL, 0, 1023, 3, 1023);

  if(goal_position < pre_position) {
   pre_position -= SPEED_VAL;
   if(goal_position > pre_position)
     pre_position=goal_position;
  }else {
   pre_position += SPEED_VAL;
   if(goal_position < pre_position)
     pre_position=goal_position;
  }

  //VR Resverse
  if(MAX_STROKE_VAL<MIN_STROKE_VAL) {
    min_stroke_limit=MAX_STROKE_VAL;
    max_stroke_limit=MIN_STROKE_VAL;
  }else {
    min_stroke_limit=MIN_STROKE_VAL;
    max_stroke_limit=MAX_STROKE_VAL;
  }

  // position Limit
  if(pre_position>=max_stroke_limit){
    pre_position = max_stroke_limit;
  }
  else if(pre_position<=min_stroke_limit){
    pre_position = min_stroke_limit;
  }

  position_val = map(pre_position, 0, 1023, 0, 180);
  Tester.servo_CH1.write(position_val);
  delay(15);
}
```

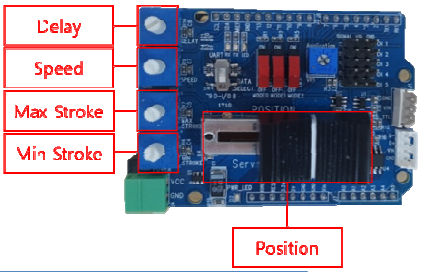
Read VR value for present position and adjust it to goal position.
According to VR value, time to approach goal position will be differentiated.

To prevent stopping condition of servo motor, adjust scale value between 3~ 1023.

Adjust present position by “SPEED_VAL” to the goal position.

5.10. Example - VR Delay (PWMControl)

Select [IRROBOT_ServoTesterShield] – [UNO] or[LEO] –[ServoTester_VR_Delay]
Control delay by Rotary volume#1.




```
#include <IRROBOT_ServoTesterShield.h>

SoftwareSerial mySerial(8,9);
IRROBOT_ServoTesterShield Tester(&mySerial);

#define APPLICATION_VR      Tester.VR_1
#define MANUAL_POSITION_VR  Tester.VR_2
#define MIN_STROKE_VR       Tester.VR_3
#define MAX_STROKE_VR       Tester.VR_4
#define SPEED_VR            Tester.VR_5
#define DELAY_VR            Tester.VR_6

int DELAY_val,DELAY_cnt;
int goal_position,pre_position;
int position_val=0;
int MIN_STROKE_VAL, min_stroke_limit;
int MAX_STROKE_VAL, max_stroke_limit ;

int SPEED_VAL;
void setup() {
  Serial.begin(9600);
  Tester.begin();
}

void loop() {
  //Reads analoge
  MIN_STROKE_VAL = MIN_STROKE_VR.read();
  MAX_STROKE_VAL = MAX_STROKE_VR.read();
  goal_position  = MANUAL_POSITION_VR.read();
  SPEED_VAL      = SPEED_VR.read();
  DELAY_val      = DELAY_VR.read();

  SPEED_VAL = map(SPEED_VAL, 0, 1023, 3, 1023);
  DELAY_val = map(DELAY_val, 0, 1023, 0, 50);

  if(DELAY_cnt>DELAY_val){
    DELAY_cnt=0;

    if(goal_position < pre_position) {
      pre_position -= SPEED_VAL;
      if(goal_position > pre_position)
        pre_position=goal_position;
    }else {
      pre_position += SPEED_VAL;
      if(goal_position < pre_position)
        pre_position=goal_position;
    }
  }else {
    DELAY_cnt++;
  }

  //VR Resverse
  if(MAX_STROKE_VAL<MIN_STROKE_VAL) {
    min_stroke_limit=MAX_STROKE_VAL;
    max_stroke_limit=MIN_STROKE_VAL;
  }else {
    min_stroke_limit=MIN_STROKE_VAL;
    max_stroke_limit=MAX_STROKE_VAL;
  }

  // position Limit
  if(pre_position>=max_stroke_limit){
    pre_position = max_stroke_limit;
  }
  else if(pre_position<=min_stroke_limit){
    pre_position = min_stroke_limit;
  }

  position_val = map(pre_position, 0, 1023, 0, 180);
  Tester.servo_CH1.write(position_val);
}
```

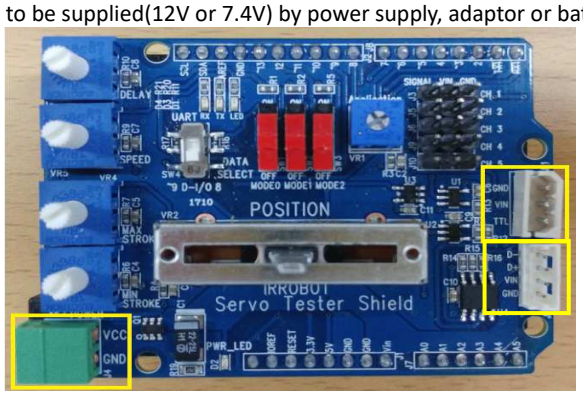
Adjust delay value by adjusting VR#1 value after reading it. According to the value of VR, time to approach goal position will be differentiated.

Adjust delay value between 0~50.

Count/update delay time in Program loop, and reset delay count.

5.11 Data Communication Control(TTL/RS-485)

IR-STS01 has each TTL(3pins) and RS-485(4pins) ports. Multiple qty of servo motors can be connected each other in serial (daisy-chain). To operate servo motor, power needs to be supplied(12V or 7.4V) by power supply, adaptor or battery.



To get feedback Echo packet for servo motor status, user needs to use Leonardo board which has two serial ports. Below example is simple packet command.

```
void setup() {
  Serial.begin(57600);
  pinMode(2, OUTPUT);
}

void loop() {
  // 서보의 목표 위치 값
  digitalWrite(2, HIGH);
  delay(1);
  Serial.write(0xff);  Serial.write(0xff);  Serial.write(0xff);
  Serial.write(0x00);
  Serial.write(0x05);
  Serial.write(0xF3);
  Serial.write(0x86);
  Serial.write(0xFF);
  Serial.write(0x07);
  int iChecksum = ~(0x00 + 0x05 + 0xF3 + 0x86 + 0xFF +0x07);
  Serial.write(iChecksum);
  Serial.flush();
  digitalWrite(2, LOW);
  delay(1000);
  // 서보의 목표 위치 값
  digitalWrite(2, HIGH);
  delay(1);
  Serial.write(0xff);  Serial.write(0xff);  Serial.write(0xff);
  Serial.write(0x00);
  Serial.write(0x05);
  Serial.write(0xF3);
  Serial.write(0x86);
  Serial.write(0x00);
  Serial.write(0x00);
  iChecksum = ~(0x00 + 0x05 + 0xF3 + 0x86 + 0x00 +0x00);
  Serial.write(iChecksum);
  Serial.flush();
  digitalWrite(2, LOW);
  delay(1000);
}
```

Command packet to assign goal position of servo motor to "2047(0x07FF)"

Command packet to assign goal position of servo motor to "0(0x0000)"

Header	0xFFFFFF
ID (모터 ID)	0x00
SIZE (패킷의 길이)	0x05
COMMAND (명령어)	0xF3
FACTOR #1(주소)	0x86
FACTOR #2(Low_Byt)	0xFF
FACTOR #3(High_Byt)	0x07
Checksum	iChecksum

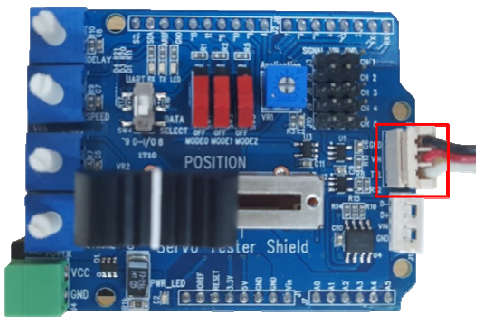
6 Servo Control Example through Arduino IDE

Control servo motor via data communication (RS-485 or TTL) on Arduino IDE. Control is available using our IR-STS01 Arduino servo tester shield or Arduino board only.

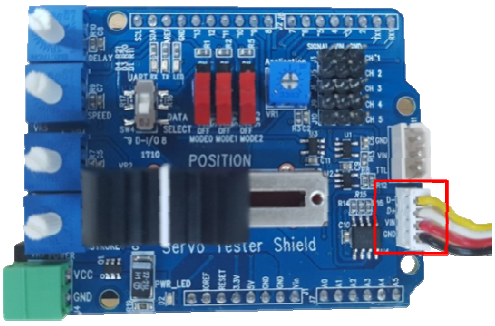
6.1 Outline

To upload example, Data select switch must be at UP position. Basic program from the factory will be removed and rewrote by user program once user uploads example. If user needs basic program again, user is able to get it from the library.

Servo motor needs to be connected with proper servo port (RS-485 and TTL) as below.



< TTL Communication Port >



< RS-485 Communication Port>

6.2. Example - Information Read (RS-485/TTL)

Example to read basic information such as ID, Version using API of MightyZap.
Select [Example] - [MightyZap] - [UNO]or[LEO]–[ServoTester_Information]

```
#include <IRROBOT_ServoTesterShield.h>
#define ID_NUM 0
IRROBOT_ServoTesterShield Tester(&Serial1);

void setup() {
  Serial.begin(9600);
  Tester.Mightyzap.begin(32);
  while (! Serial);
}

void loop() {
  if(Serial.available())  {
    char ch = Serial.read();
    if(ch=='d')    {
      Serial.print("Model Number        : ");  Serial.println(Tester.Mightyzap.getModelNumber(ID_NUM));
      Serial.print("Firmware Version    : ");  Serial.println(Tester.Mightyzap.Version(ID_NUM));
      Serial.print("CAL Short Stroke    : ");  Serial.println(Tester.Mightyzap.CalStroke(ID_NUM,Short));
      Serial.print("CAL Long Stroke     : ");  Serial.println(Tester.Mightyzap.CalStroke(ID_NUM,Long));
      Serial.print("CAL Center Stroke   : ");  Serial.println(Tester.Mightyzap.CalStroke(ID_NUM,Center));
      Serial.print("Present Volt        : ");  Serial.println(Tester.Mightyzap.presentVolt(ID_NUM)/10);
      Serial.print("Present Temperature : ");  Serial.println(Tester.Mightyzap.presentTemperature(ID_NUM));
    }
  }
}
```

.

#deifne ID_NUM / servo ID Assignment

- Function : Assign servo ID number for control.
- Servo Motor ID
  - 0 : Default setting value. Stand-alone ID(hereinafter, 0 will be used as an ID for all examples.)
  - 1 ~ 253 : Individual ID from 1 to 253.
  - 254 : Broadcasting ID. All different ID servos to be operated with ID254. However, Feedback Packet does not operate.

IRROBOT_ServoTesterShield Tester(&serial)/ Communication setting when use of MightyZap Sheild IR-STS01

- Function : Communication port setting with servo motor
- Parameter
  - Serial : when Arduino Leonardo Board is used, communication with servo motor to be made via serial port1.

Tester.Mightyzap.begin(int baudrate)/Communication speed setting

- Function : Servo motor communication speed setting
- Parameter
  - ID_NUM : Servo motor ID
  - Baudrate : Set communication speed

Data	Description
16	115200 baudrate
32	57600 baudrate
64	19200 baudrate
128	9600 baudrate

Tester.Mightyzap.getModelNumber(nt ID_NUM)/Parameter Read

- Function : Servo motor model number check

- Parameter - ID_NUM : Servo motor ID

Tester.Mightyzap.Virsion(nt ID_NUM)/Parameter Read

- Function : Servo motor version check
- Parameter - ID_NUM : Servo motor ID

Tester.Mightyzap.CalStroke(nt ID_NUM,int Direction)/Parameter Read

- Function : Servo motor's stroke position calibration value check
- Parameter

- ID_NUM : Servo motor ID
- Direction

Data	Description
Short	Short Stroke position calibration value, save factory default value
Long	Long Stroke position calibration value, save factory default value
Center	Center Stroke position calibration value, save factory default value.

Tester.Mightyzap.PresentVolt(nt ID_NUM)/Parameter Read

- Function : Servo motor's current input voltage check
- Parameter - ID_NUM Servo motor ID

Tester.Mightyzap.PresentTemperature(nt ID_NUM)/Parameter Read

- Function : Servo motor's current temperature check (temperature is not exact info, so use this info just for reference.)
- Parameter - ID_NUM : Servo motor ID

6.3. Example - Servo ID

ID setting of servo motor using ServoID() command.
Select [Example] - [MightyZap] - [UNO]or[LEO]–[ServoTester_ ServoID]

```
#include <IRROBOT_ServoTesterShield.h>
#define ID_NUM 0
IRROBOT_ServoTesterShield Tester(&Serial1);

int ID_Sel =0;
void setup() {
  Serial.begin(9600);
  Tester.Mightyzap.begin(32);
  while (! Serial);
  Serial.print("Input ID : ");
}

void loop() {
  Tester.Mightyzap.ledOn(1,RED);
  Tester.Mightyzap.ledOn(2,GREEN);
  Tester.Mightyzap.ledOn(3,BLUE);

  if(Serial.available()) {
    ID_Sel = Serial.parseInt();
    Serial.println(ID_Sel);
    Serial.print("Input_ID [0~3] : ");
    Tester.Mightyzap.ServoID(0,ID_Sel);
    Tester.Mightyzap.ServoID(1,ID_Sel);
    Tester.Mightyzap.ServoID(2,ID_Sel);
    Tester.Mightyzap.ServoID(3,ID_Sel);
  }
}
```

Control LED color using LedON() command.
After changing ID of servo motor, check ID change status by assign different LED color for changed ID.

As All examples are using ID 0, so change ID to 0 again.

Tester.Mightyzap.ServoID(int ID_NUM, int ID_SET)/Parameter Read

- Function : Servo motor ID setting
- Parameter

- ID_NUM : Servo motor ID
- ID_SET : Servo motor's amended ID

6.4. Example - LED

Setting LED color using LedON() command.
Select [Example] - [MightyZap] - [UNO]or [LEO]– [ServoTester_ LED]

```
#include <IRROBOT_ServoTesterShield.h>
#define ID_NUM 0
IRROBOT_ServoTesterShield Tester(&Serial1);

void setup() {
  Tester.Mightyzap.begin(32);
}
void loop() {
  Tester.Mightyzap.ledOn(ID_NUM,RED);
  delay(1000);
  Tester.Mightyzap.ledOn(ID_NUM,GREEN);
  delay(1000);
  Tester.Mightyzap.ledOn(ID_NUM,BLUE);
  delay(1000);
}
```

Change LED color using LedON() command.
Use RED, GREEN, BLUE to express various color.
LED alarm is the highest priority.

Tester.Mightyzap.LED(int ID_NUM, int Color)/Parameter Write

- Function : Servo motor LED setting
- Parameter

- ID_NUM : Servo Motor ID
- Color : Servo motor LED Color

Data	Description
RED	RED ON
GREEN	GREEN ON
BLUE	BLUE ON

6.5. Example - Limit Temperature

Check/setting temperature limit using LimitTemperature() command.
Select [Example] - [MightyZap] - [ServoTester_UNO/LEO]–[ServoTester_ LimitTemperature]

```
#include <IRROBOT_ServoTesterShield.h>
#define ID_NUM 0
IRROBOT_ServoTesterShield Tester(&Serial1);

int Temperature;
void setup() {
  Serial.begin(9600);
  Tester.Mightyzap.begin(32);
  while (! Serial);
  Serial.print("*LimitTemperature : ");
  Serial.println(Tester.Mightyzap.maxTemperature(ID_NUM));
}

void loop() {
  if(Serial.available()) {
    Temperature = Serial.parseInt();
    Tester.Mightyzap.maxTemperature(ID_NUM,Temperature);
    Serial.print("*LimitTemperature : ");
    Serial.println(Tester.Mightyzap.maxTemperature(ID_NUM));
    Serial.print("*presentTemperature : ");
    Serial.println(Tester.Mightyzap.presentTemperature(ID_NUM));
  }
}
```

Set temp limit using LimitTemperature() command.

In case of out of temp limit, servo motor to be operated according to the alarm setting.

Tester.Mightyzap.limitTemperature(int ID_NUM)/Parameter Read

- Function : Read max/min temperature limit
- Parameter
  - ID_NUM : Servo motor ID

Tester.Mightyzap.limitTemperature (int ID_NUM,int Limit)/Parameter Write

- Function : Write max/min temperature limit
- Parameter
  - ID_NUM : Servo motor ID
  - Limit : Servo motor Operation temp limit setting

6.6. Example - Goal Position

Example to control servo motor position using goalPosition() command.
Select [Example] - [MightyZap] - [UNO]or[LEO]–[ServoTester_ goalPosition]

```
#include <IRROBOT_ServoTesterShield.h>
#define ID_NUM 0
IRROBOT_ServoTesterShield Tester(&Serial1);
int ForceLimit;

void setup() {
  Tester.Mightyzap.begin(32);
}

void loop() {
  Tester.Mightyzap.goalPosition(ID_NUM, 0); //ID (
  delay(1000);
  Tester.Mightyzap.goalPosition(ID_NUM, 4095);//ID
  delay(1000);
}
```

By 1sec interval, control position between 0 ~ 4095.
Change position value in goalPosition(ID_NUM,0) function and check position difference.

Tester.Mightyzap.goalPosition(int ID_NUM, int postion)/Parameter Write

- Function : Servo motor goal position setting
- Parameter
  - ID_NUM : Servo motor ID
  - Position : Goal position value between 0~4095 (defualt : 0~4095)

6.7. Example - Present Position

Example to check current position of servo motor using presentPosition() command.
Select [Example] - [MightyZap] - [UNO]or[LEO]–[ServoTester_ presentPosition]

```
#include <IRROBOT_ServoTesterShield.h>
#define ID_NUM 0
IRROBOT_ServoTesterShield Tester(&Serial1);

int Position, cPosition, Display =1;

void setup() {
  Serial.begin(9600);
  Tester.Mightyzap.begin(32);
  while (! Serial);
  Tester.Mightyzap.movingSpeed(ID_NUM,100);
}

void loop() {
  if(Display == 1){
    Serial.print("*New Position[0~4095] : ");
    Display = 0;
  }
  if(Serial.available())  {
    Position = Serial.parseInt();
    Serial.println(Position);
    delay(200);

    Tester.Mightyzap.goalPosition(ID_NUM,Position);
    delay(200);
    while(Tester.Mightyzap.Moving(ID_NUM)) {
      cPosition = Tester.Mightyzap.presentPosition(ID_NUM);
      Serial.print("  - Position : ");
      Serial.println(cPosition);
    }
    delay(200);
    cPosition = Tester.Mightyzap.presentPosition(ID_NUM);
    Serial.print("  - final Position : ");
    Serial.println(cPosition);
    Display = 1;
  }
}
```

After changing position using goalPosition() command, check real-time present position by presentPosition() command.

*Moving() : to check operation status (no moving or moving)
(1 : Moving, 0: Moving Stop)

Tester.Mightyzap.presentPosition(int ID_NUM)/Parameter Read

- Function : Servo motor position setting
- Parameter
  - ID_NUM : Servo motor ID

Tester.Mightyzap.moving(int ID_NUM)/Parameter Read

- Function : Servo motor Operation stutus read.
- Parameter
  - ID_NUM : Servo motor ID

6.8. Example - Limit Volt

Check/setting input voltage limit by LimitVolt() command.
Select [Example] - [MightyZap] - [UNO]or[LEO]–[ServoTester_ LimitVolt]

```
include <IRROBOT_ServoTesterShield.h>
#define ID_NUM 0
IRROBOT_ServoTesterShield Tester(&Serial1);

int LimitVolt;
int Display = 1;

void setup() {
  Serial.begin(9600);
  Tester.Mightyzap.begin(32);
  Serial.print("*Limit Lowest Volt : ");
  Serial.println(Tester.Mightyzap.limitVolt(ID_NUM,Lowest));
  while (! Serial);
}

void loop() {
  if(Display == 1){
    Serial.print("*Limit Lowest Volt : ");
    Serial.println(Tester.Mightyzap.limitVolt(ID_NUM,Lowest));
    Display = 0;
  }
  if(Serial.available()) {
    LimitVolt = Serial.parseInt();
    Tester.Mightyzap.limitVolt(ID_NUM,Lowest,LimitVolt);
    Display = 1;
  }
}
```

Check/setting input voltage max/min limit by LimitVolt() command.

Change “Lowest” to “Highest” in LimitVolt(IN_NUM, Lowest, LimitVolt) to change voltage limit setting

In case of out of limit input voltage, servo to be operated according to alarm setting.

Tester.Mightyzap.limitVolt(int ID_NUM,Int Direction)/Parameter Read

- Function : max/min voltage read
- Parameter
  - ID_NUM : Servo motor ID
  - Direction : max/min voltage select

Data	Description
Lowest	Max(highest) voltage read
highest	Min(lowest) voltage read

Tester.Mightyzap.limitVolt(int ID_NUM,Int Direction,int Limit)/Parameter Write

- Function : max/min voltage write
- Parameter
  - ID_NUM : Servo motor ID
  - Direction : max/min voltage setting

Data	Description
Lowest	Select Min(lowest) voltage select
highest	Max(highest) voltage select

  - Limit : Operation limit voltage setting

6.9. Example - Alarm LED

Setting LED Alarm indicator by AlarmLed()
Select [Example] - [MightyZap] - [UNO]or[LEO]–[ServoTester_ AlarmLed]

```
#include <IRROBOT_ServoTesterShield.h>
#define ID_NUM 0
IRROBOT_ServoTesterShield Tester(&Serial1);
int alarm;

void setup() {
  Serial.begin(9600);
  Tester.Mightyzap.begin(32);
  while (! Serial);
  Serial.print("*Alarm LED Setting : ");
  Serial.println(Tester.Mightyzap.alarmLed(ID_NUM));
  Serial.print("*Alarm LED Setting : ");
}

void loop() {
  if(Serial.available()) {
    alarm = Serial.parseInt();
    Tester.Mightyzap.alarmLed(ID_NUM,alarm);
    Serial.println(Tester.Mightyzap.alarmLed(ID_NUM));
    Serial.print("*Alarm LED State : ");
  }
}
```

alarmLed()
- When Error occurs, LED will be ON if bit is set at 1. (1=enable / 0=disable)

Example shows flickering LED operation and low bit has a priority when different errors occur at the same time.

After error condition is resolved, alarm will be inactivated after 2 sec and return to normal status.

Tester.Mightyzap.alarmLed(int ID_NUM)/Parameter Read

- Function : Servo motor LED read
- Parameter
  - ID_NUM : Servo motor ID

Tester.Mightyzap.alarmLed(int ID_NUM,Int Alarm)/Parameter Write

- Function : Servo motor LED write
- Parameter
  - ID_NUM : Servo motor ID
  - Alarm : Servo motor Alarm LED setting

Error	Bit
RESERVED	7
Instruction Error	6
Overload Error	5
Checksum error	4
Range Error	3
Stroke Limit Error	1
Input Voltage Error	0

6.10. Example - AlarmShutdown

Example for alarm shutdown(power cut off when error occurs) using AlarmShutdown() command.
Select [Example] - [MightyZap] - [UNO] or [LEO]–[ServoTester_ AlarmShutdown]

```
#include <IRROBOT_ServoTesterShield.h>
#define ID_NUM 0

IRROBOT_ServoTesterShield Tester(&Serial1);

int alarm;
int Display = 1;

void setup() {
  Serial.begin(9600);
  Tester.Mightyzap.begin(32);
  while (! Serial);
  Serial.print("*Alarm Shutdown State : ");
  Serial.println(Tester.Mightyzap.alarmShutdown(ID_NUM));
  Serial.print("*Alarm Shutdown State : ");
}

void loop() {
  if(Serial.available()) {
    alarm = Serial.parseInt();
    Tester.Mightyzap.alarmShutdown(ID_NUM,alarm);
    Serial.println(Tester.Mightyzap.alarmShutdown(ID_NUM));
    Serial.print("*Alarm Shutdown State : ");
  }
}
```

alarmShutdown()
When error occurs, power will be off(servo shutdown) to protect servo motor. Setting bit as 1, this function will be enabled. (1=enable / 0=disable)

After alarm shutdown setting using alarmShutdown() function, make an error condition on purpose using other command which may make an error.

Tester.Mightyzap.alarmShutdown(int ID_NUM)/Parameter Read

- Function : Servo motor alram shutdown read
- Parameter
  - ID_NUM : Servo motor ID

Tester.Mightyzap.alarmShutdown(int ID_NUM,Int Alarm)/Parameter Write

- Function : Servo motor alram shutdown write
- Parameter
  - ID_NUM : Servo motor ID
  - Alarm : Servo motor Alarm Shutdown setting

Error	Bit
RESERVED	7
Instruction Error	6
Overload Error	5
Checksum Error	4
Range Error	3
Stroke Limit Error	1
Input Voltage Error	0

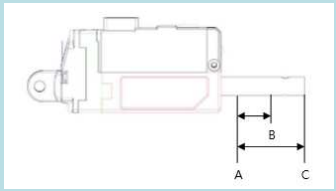
6.11.Example - Stroke Limit

Example to set stroke limit(limit for position) using StrokeLimit() command.
Select [Example] - [MightyZap] - [UNO] or[LEO]–[ServoTester_ StrokeLimit]

```
#include <IRROBOT_ServoTesterShield.h>
#define ID_NUM 0

IRROBOT_ServoTesterShield Tester(&Serial1);

void setup() {
  Tester.Mightyzap.begin(32);
}

void loop() {
  Tester.Mightyzap.StrokeLimit(ID_NUM,Long,3095);
  Tester.Mightyzap.goalPosition(ID_NUM,4095);
  delay(1000);

  Tester.Mightyzap.StrokeLimit(ID_NUM,Short,100);
  Tester.Mightyzap.goalPosition(ID_NUM,0);
  delay(1000);
}
```

1.StrokeLimit()
- Setting max(long)/min(short) position limit to control servo motor position's limit.
  - User may decrease max(long) stroke limit or increase min(short) stroke limit.

2. Shot Stroke(A)
   Long Stroke(C)



Tester.Mightyzap .StrokeLimit(int ID_NUM ,int Direction)/Parameter Read

- Function : Servo Motor position limit read
- Parameter
  - ID_NUM : Servo motor ID
  - Direction : Direction of Servo motor (Long or short)

Tester.Mightyzap .StrokeLimit(int ID_NUM,int Direction, int position)/Parameter Write

- Function : Servo motor position limit setting
- Parameter
  - ID_NUM : Servo motor ID
  - Direction : Servo motor position direction setting

Data	Bit
Short	Retract(min) limit position
Long	Extend(max) limit position

  - Position : Limit position value setting (default : 0~4095)

6.12.Example - Resolution Factor

Example to set Resolution of servo motor position using ResolutionFactor() command.
Select [Example] - [MightyZap] - [UNO/LEO]–[ServoTester_ ResolutionFactor]

```
#include <IRROBOT_ServoTesterShield.h>
#define ID_NUM 0
IRROBOT_ServoTesterShield Tester(&Serial1);
int Resolution;
int Display=1;

void setup() {
  Serial.begin(9600);
  Tester.Mightyzap.begin(32);
  while (! Serial);
}

void loop() {
  if(Display==1){
    Tester.Mightyzap.goalPosition(ID_NUM,512);
    delay(1000);
    Tester.Mightyzap.goalPosition(ID_NUM,0);
    delay(1000);
    //  Factor 1: Resolution 4096
    //  Factor 2: Resolution 2048
    //  Factor 3: Resolution 1024
    //  Factor 4: Resolution 512
    Serial.print("Input Resolution Factor(1~4) : ");
    Display=0;
  }
  if(Serial.available()) {
    Resolution = Serial.parseInt();
    Tester.Mightyzap.resolutionFactor(ID_NUM,Resolution);
    Serial.println(Resolution);
    delay(500);
    Display=1;
  }
}
```

- ResolutionFactor()
  Change positional resolution of Servo motor
  to 512/1024/2048/4096 (default : 4096)
- The Higher resolution, the higher positional accuracy

Resolution factor can be changed by serial communication.

Tester.Mightyzap.ResolutionFactor(int ID_NUM)/Parameter Read

- Function : Servo motor resolution read
- Parameter

- ID_NUM : Servo motor ID

Tester.Mightyzap.ResolutionFactor(int ID_NUM,int ResolutionFactor)/Parameter Write

- Function : Servo motor resolution setting
- Parameter

- ID_NUM : Servo motor ID
- ResolutionFactor: Servo motor position resolution

Data	Description
1	4096 (Setting positional resolution to 0~4096)
2	2048 (Setting positional resolution to 0~2048)
3	1024 (Setting positional resolution to 0~1024)
4	512 (Setting positional resolution to 0~512)

6.13. Moving Speed

Example to set servo motor speed using MovingSpeed() command.
Select [Example] - [MightyZap] - [UNO/LEO]–[ServoTester_ MovingSpeed]

```
#include <IRROBOT_ServoTesterShield.h>
#define ID_NUM 0
IRROBOT_ServoTesterShield Tester(&Serial1);
int Speed, Display=1;

void setup() {
  Serial.begin(9600);
  Tester.Mightyzap.begin(32);
  while (! Serial);
}

void loop() {
  if(Display ==1) {
    Tester.Mightyzap.goalPosition(ID_NUM,0);
    delay(500);
    while(Tester.Mightyzap.Moving(ID_NUM));

    Tester.Mightyzap.goalPosition(ID_NUM,4095);
    delay(500);
    while(Tester.Mightyzap.Moving(ID_NUM));
    Serial.print("New Speed[0~1023] : ");
    Display=0;
  }
  if(Serial.available()) {
    Speed = Serial.parseInt();
    Tester.Mightyzap.movingSpeed(ID_NUM,Speed);
    Serial.println(Tester.Mightyzap.movingSpeed(ID_NUM));
    delay(500);
    Display=1;
  }
}
```

1.movingSpeed()
- Set moving speed.
- 0 : Max speed
- 1~1023 : the higer value, the faster speed.
- To be initialized to 0 when Power is applied again.

2.Check operation status by moving() function.
(1 : moving, 0: not moving)

3. Change servo motor speed by serial communication and check its motion.

Tester.Mightyzap.movingSpeed(int ID_NUM)/Parameter Read

- Function : Servo motor moving speed read
- Parameter
  - ID_NUM : Servo motor ID

Tester.Mightyzap.movingSpeed(int ID_NUM,int speed)/Parameter Write

- Function : Servo motor moving speed value setting
- Parameter
  - ID_NUM : Servo motor ID
  - Speed: Setting servo speed between 0 ~ 1023 (0:Max speed, 1~1023 : the higher value, the faster speed)

6.14 Force Limit

Example to set Force limit by ForceLimit() command.
Select [Example] - [MightyZap] - [UNO/LEO]–[ServoTester_ ForceLimit]

```
#include <IRROBOT_ServoTesterShield.h>
#define ID_NUM 0
IRROBOT_ServoTesterShield Tester(&Serial1);

int ForceLimit;

void setup() {
  Serial.begin(9600);
  Tester.Mightyzap.begin(32);
  while (! Serial);
   Serial.print("* Max Force[0~1023] : ");
}

void loop() {
  Tester.Mightyzap.goalPosition(ID_NUM,0);
  delay(1000);
  Tester.Mightyzap.goalPosition(ID_NUM,4015);
  delay(1000);

  if(Serial.available()) {
    ForceLimit = Serial.parseInt();
    Tester.Mightyzap.forceLimit(ID_NUM,ForceLimit);
    Serial.println(ForceLimit);
    Serial.print("* Max Force[0~1023] : ");
  }
}
```

ForceLimit()
- Servo motor force limit setting
- Set between 0~1023. The higher value, the stronger force.
- when Power is applied again, Max force value will be copied to Force limit value.

Change force by ForceLimit() function and check operation status. The lower force may make slower operation.

Tester.Mightyzap.forceLimit(int ID_NUM)/Parameter Read

- Function : Servo motor force limit read
- Parameter
  - ID_NUM : Servo motor ID

Tester.Mightyzap.maxForce(int ID_NUM, int Force)/Parameter Write

- Function : Servo motor force limit setting
- Parameter
  - ID_NUM : Servo motor ID
  - Force : Set Servo motor force between 0 ~ 1023

6.15. Max Force

Example to set Max force by using MaxForce() command.
Select [Example] - [MightyZap] - [UNO/LEO]–[ServoTester_MaxForce]
The function of Max force is same as Force limit described previously. But Max force is non- volatile value, so it will not be removed even after power off while Force limit is a volatile parameter and will be removed when power is off. Thus, Max force setting will be copied to Force limit value when the servo reboots.

```
#include <IRROBOT_ServoTesterShield.h>
#define ID_NUM 0
IRROBOT_ServoTesterShield Tester(&Serial1);

int MaxForce;
int ForceLimit;
int Display=1;

void setup() {
  Serial.begin(9600);
  Tester.Mightyzap.begin(32);
  while (! Serial);
}

void loop() {
f(Display==1){
    MaxForce = Tester.Mightyzap.maxForce(ID_NUM);
    Serial.print("Max Force   : ");
    Serial.println(MaxForce);
    ForceLimit = Tester.Mightyzap.forceLimit(ID_NUM);
    Serial.print("Force Limit : ");
    Serial.println(ForceLimit);
    Serial.println("-------------------------");
    Display =0;
  }
  if(Serial.available()) {
    MaxForce = Serial.parseInt();
    Tester.MightyzapMax.maxForce(ID_NUM,MaxForce);
    Display =1;
   }
}
```

MaxForce()
- Setting/check Max force of Servo motor.
  Set between 0~1023. The higher value, the stronger force.

The difference from Force limit is that Max force is a non-volatile value while force limit is a volatile value. When power is applied again, Max force value will be copied to Force limit value.

Change max force using MaxForce() function and check how it works with force limit.

Tester.Mightyzap.maxForce(int ID_NUM)/Parameter Read

- Function : Servo motor max force read
- Parameter
  - ID_NUM : Servo motor ID

Tester.Mightyzap.maxForce(int ID_NUM, int Force)/Parameter Write

- Function : Servo motor max force setting
- Parameter
  - ID_NUM : Servo motor ID
  - Force : Set Max Force value between 0 ~ 1023

6.16. Compliance Margin

Set maximum positional error allowance using ComplianceMargin() command.
Select [Example] - [MightyZap] - [UNO/LEO]–[ServoTester_ComplianceMargin]

```
#include <IRROBOT_ServoTesterShield.h>
#define ID_NUM 0
IRROBOT_ServoTesterShield Tester(&Serial1);

int Margin, Position, Display=1;

void setup() {
  Serial.begin(9600);
  Tester.Mightyzap.begin(32);
  while (! Serial);
}

void loop() {
  if(Display == 1) {
    Tester.Mightyzap.goalPosition(ID_NUM,0);
    delay(1000);
    Position = Tester.Mightyzap.presentPosition(ID_NUM);
    Serial.print("  - Short Position : ");
    Serial.println(Position);

    Tester.Mightyzap.goalPosition(ID_NUM,4095);
    delay(1000);
    Position = Tester.Mightyzap.presentPosition(ID_NUM);
    Serial.print("  - Long Position : ");
    Serial.println(Position);
    Serial.println(" ");
    Serial.print("* New Compliance Margin : ");
    Display = 0;
  }
  if(Serial.available()) {
     Margin = Serial.parseInt();
    Tester.Mightyzap.complianceMargin(ID_NUM,Short,Margin);
    Tester.Mightyzap.complianceMargin(ID_NUM,Long,Margin);
    Serial.println(Margin);
    Display = 1;
  }
}
```

complianceMargin()

- Set maximum positional error allowance
- The higher compliance margin, the higher deadband. The lower compliance margin, the higher accuracy, but the higher chance of jitter.

Compliance margin can be set for each retract(short) and extend(long) direction.

Check position error range using goalPosition() command.

Tester.Mightyzap.complianceMargin(int ID_NUM,int Direction)/Parameter Read

- Function : Compiance Margin read
- Parameter

- ID_NUM : Servo motor ID
- Direction

Data	Description
Short	Compiance Margin for retract(short) direction
Long	Compiance Margin for extend(long) direction

Tester.Mightyzap.complianceMargin(int ID_NUM, int Direction, int margin)/Parameter Write

- Function : Compiance Margin setting
- Parameter

- ID_NUM : Servo motor ID
- Direction

Data	Description
Short	Compiance Margin for retract(short) direction
Long	Compiance Margin for extend(long) direction

- Margin : Compliance margin means error range between goal position and present position. The higher compliance margin, the higher deadband. The lower compliance margin, the higher accuracy, but the higher chance of jitter.

6.17. Punch

Set current amount of servo motor using Punch() command.
Select [Example] - [MightyZap] - [UNO/LEO]–[ServoTester_Punch]

```
#include <IRROBOT_ServoTesterShield.h>
#define ID_NUM 0
IRROBOT_ServoTesterShield Tester(&Serial1);

int punch_data;
int Display = 0;

void setup() {
  Serial.begin(9600);
  Tester.Mightyzap.begin(32);
  while (! Serial);
  Serial.print("Present Punch : ");
  Serial.println(Tester.Mightyzap.Punch(ID_NUM));
  Serial.print("Input Punch : ");
}

void loop() {
  if(Display ==1 ){
    Tester.Mightyzap.goalPosition(ID_NUM,0);
    delay(1000);
    Tester.Mightyzap.goalPosition(ID_NUM,4095);
    delay(1000);
    Serial.print("Input Punch : ");
    Display = 0;
  }
  if(Serial.available()) {
    punch_data = Serial.parseInt();
    Tester.Mightyzap.Punch(ID_NUM,punch_data);
    Serial.println(Tester.Mightyzap.Punch(ID_NUM));
    delay(500);
    Display = 1;
  }
}
```

m_zap.Punch()

Punch is a minimum current for servo motor operation. The higher punch value, the higher stall torque, but the higher chance of jitter if it is too high.

Check minimum operating current of servo motor using Punch() function.

Control minimum operating current of servo motor using Punch() function.Punch() and check difference of servo motor operation.

Tester.Mightyzap.Punch(int ID_NUM)/Parameter Read

- Function : Servo motor Punch value (min operating current value) read
- Parameter
  - ID_NUM : Servo motor ID

Tester.Mightyzap.Punch(int ID_NUM, int Punch)/Parameter Write

- Function : Servo motor Punch value (min operating current value) setting
- Parameter
  - ID_NUM : Servo motor ID
  - Punch : Min. current for motor operation.

6.18. Punch Initial

Set initial current of servo motor using PunchInitial() command.
Select [Example] - [MightyZap] - [UNO/LEO]–[ServoTester_PunchInitial]

```
#include <IRROBOT_ServoTesterShield.h>
#define ID_NUM 0
IRROBOT_ServoTesterShield Tester(&Serial1);

int PunchInitial;
int Punch;
int Display=1;

void setup() {
  Serial.begin(9600);
  Tester.Mightyzap.begin(32);
  while (! Serial);
}

void loop() {
  if(Display==1){
    PunchInitial = Tester.Mightyzap.PunchInitial(ID_NUM);
    Serial.print("Punch Initial Value: ");
    Serial.println(PunchInitial);
    Punch = Tester.Mightyzap.Punch(ID_NUM);
    Serial.print("Punch : ");
    Serial.println(Punch);
    Display =0;
  }
  if(Serial.available()) {
    PunchInitial = Serial.parseInt();
    Tester.Mightyzap.PunchInitial(ID_NUM,PunchInitial);
    Display=1;
  }
}
```

PunchInitial()
- Set punch initial value.

Punch initial value will be saved at Punch parameter when power is applied to servo motor.

After changing Punch initial value, Check saved Punch initial value by applying power again (reboot).

Tester.Mightyzap.PunchInitial(int ID_NUM)/Parameter Read

- Function : Servo motor Punch(min operating current value) initial value read
- Parameter
  - ID_NUM : Servo motor ID

Tester.Mightyzap.PunchInitial(int ID_NUM, int Punch)/Parameter Write

- Function : Servo motor Punch(min operating current value) initial value setting
- Parameter
  - ID_NUM : Servo motor ID
  - Punch : Punch Initial Data

6.19. PID

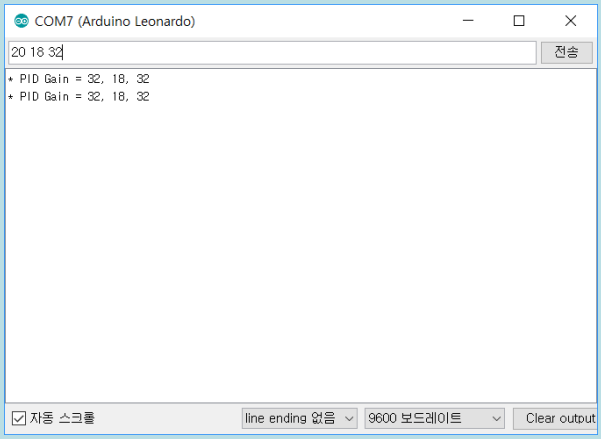
Set optimal operation of servo motor using PID command.
Select [Example] - [MightyZap] - [UNO/LEO]–[ServoTester_PID]

```
#include <IRROBOT_ServoTesterShield.h>
#define ID_NUM 0
IRROBOT_ServoTesterShield Tester(&Serial1);
int pData,iData,dData;
int Display=1;

void setup() {
  Serial.begin(9600);
  Tester.Mightyzap.begin(32);
  while (! Serial);
}

void loop() {
   if(Display==1){
    Serial.print("* PID Gain = ");
    pData = Tester.Mightyzap.pidGain(ID_NUM,pGain);
    iData = Tester.Mightyzap.pidGain(ID_NUM,iGain);
    dData = Tester.Mightyzap.pidGain(ID_NUM,dGain);
    Serial.print(pData);  Serial.print(", ");
    Serial.print(iData);  Serial.print(", ");
    Serial.println(dData);

    Tester.Mightyzap.goalPosition(ID_NUM,4095);
    delay(1000);
    Tester.Mightyzap.goalPosition(ID_NUM,0);
    delay(1000);
    Display = 0;
  }
  if(Serial.available()) {
    pData = Serial.parseInt();
    iData = Serial.parseInt();
    dData = Serial.parseInt();
    Tester.Mightyzap.pidGain(ID_NUM,pGain,pData);
    Tester.Mightyzap.pidGain(ID_NUM,iGain,iData);
    Tester.Mightyzap.pidGain(ID_NUM,dGain,dData);
    Display = 1;
  }
}
```

pidGain()
- Set each PID value of Servo motor

After setting each PID value, check the difference of operation.

Put each PID value with a space in Serial window as below.
20 18 32



Tester.Mightyzap.PID(int ID_NUM, int PID)/Parameter Read

- Function : Servo motor PID value read
- Parameter
  - ID_NUM : Servo motor ID
  - PID : pGain, iGain, dGain;

Tester.Mightyzap.PID(int ID_NUM, int PID, int Gain)/Parameter Write

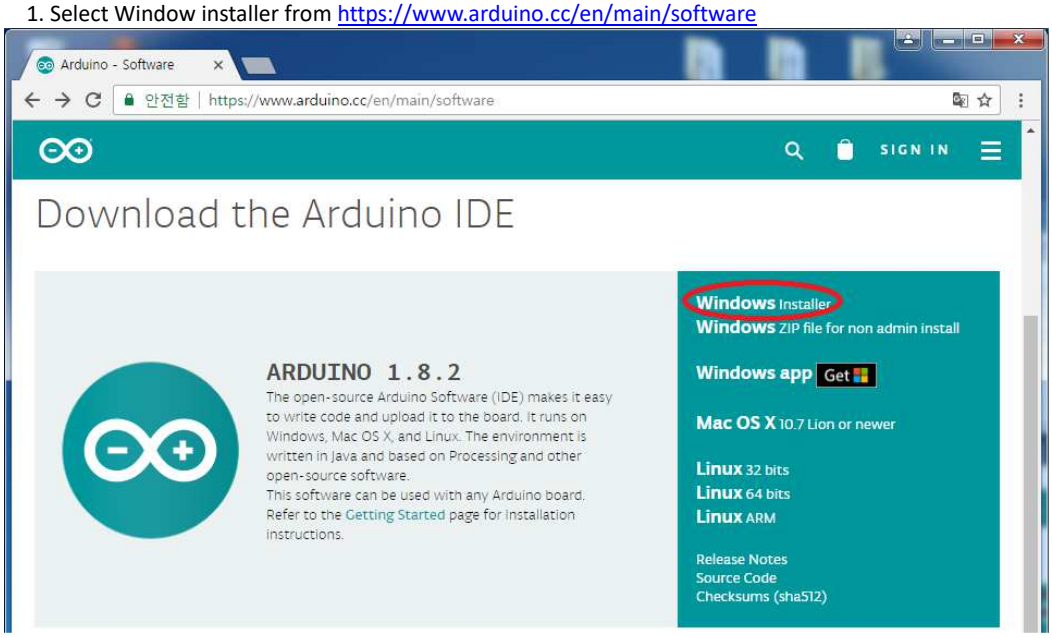
- Function : Servo motor PID setting
- Parameter
  - ID_NUM : Servo motor ID
  - PID : pGain, iGain, dGain
  - Gain : Each PID Gain value

[Appendix - Arduino PC Development Environment Setting]

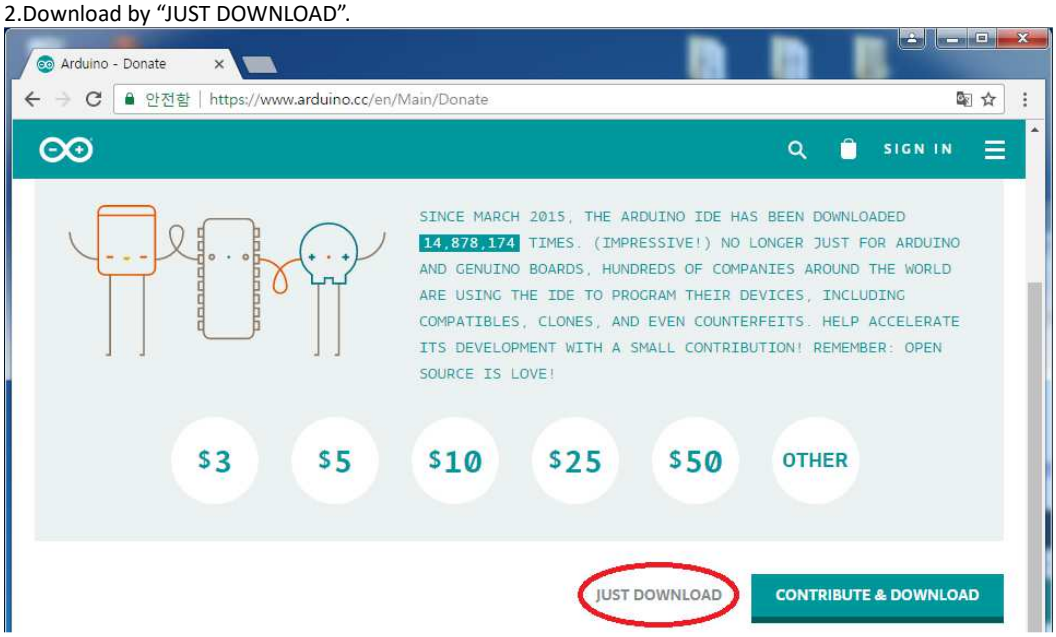
IR-STS01, Arduino Servo Tester Shield can be used with Arduino Uno or Arduino Leonardo. User needs to set Arduino development environment as below for program operation.

1 Install Arduino IDE

1. Select Window installer from https://www.arduino.cc/en/main/software

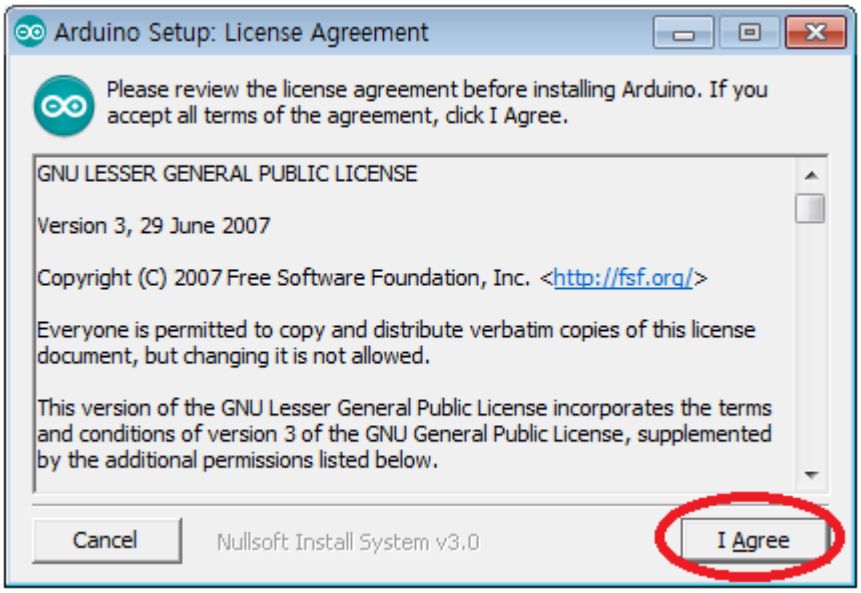


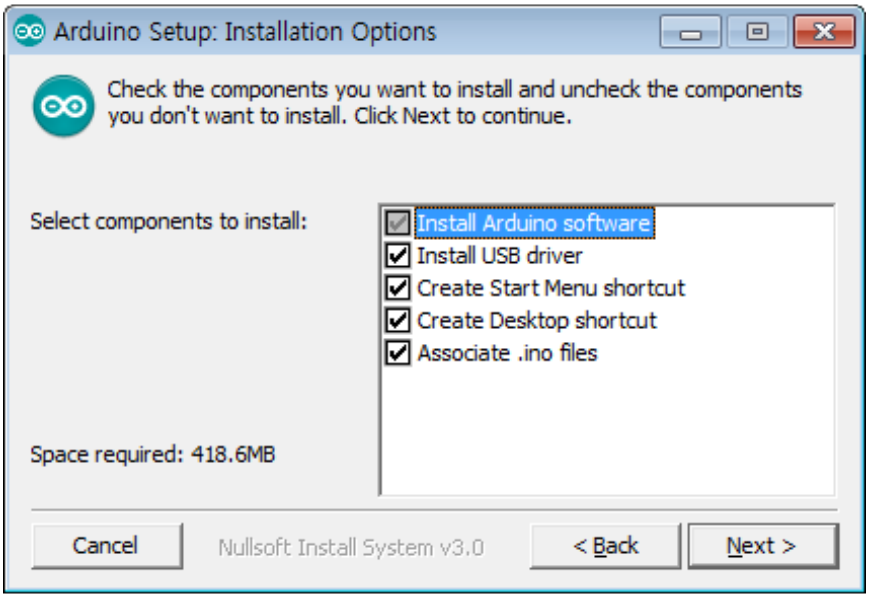
2.Download by “JUST DOWNLOAD”.

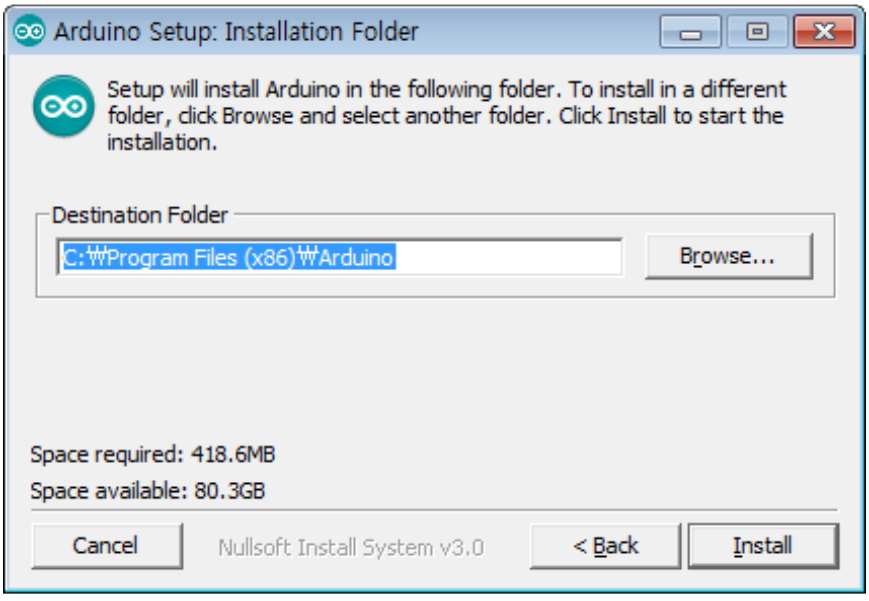


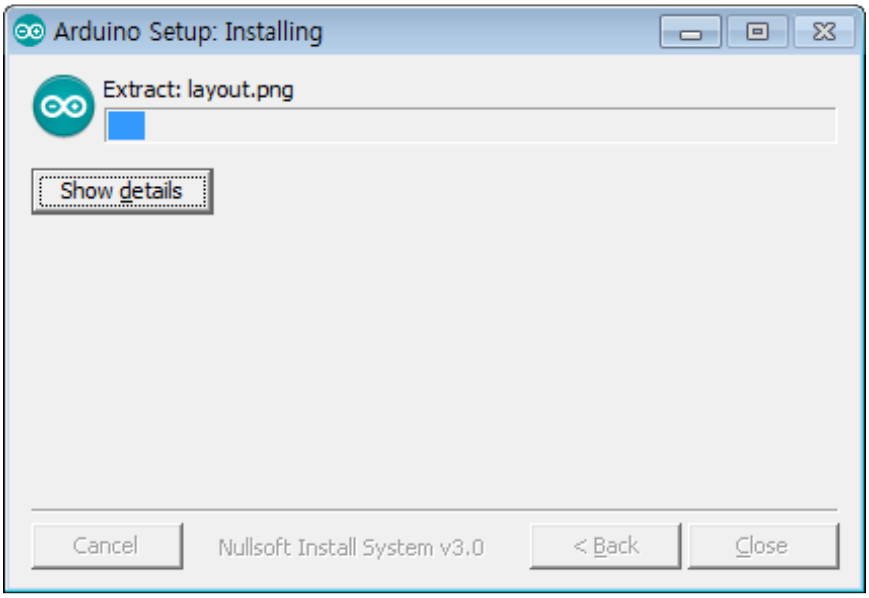
3. Run arduino-1.8.2.exe after download.

4. Install software as following.

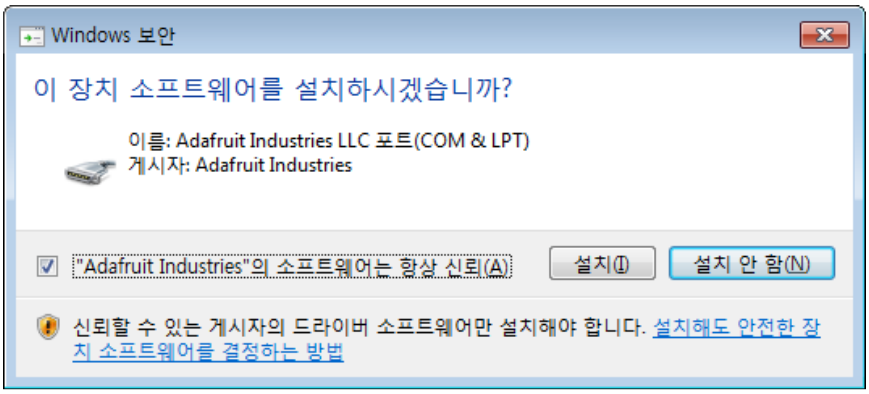




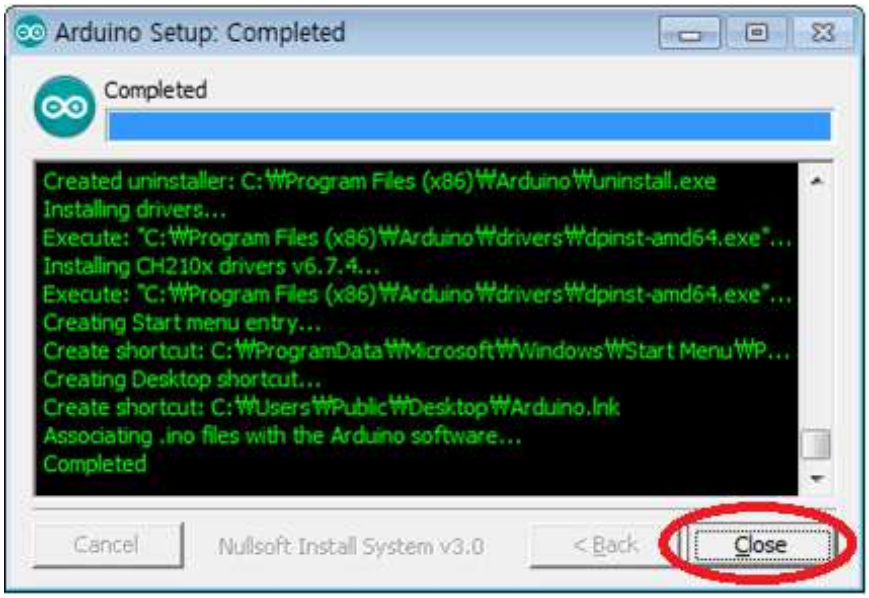




5. Install driver when Windows asks it.



6. Click “Close” after installation.



7. Run “Arduino” on your Windows wallpaper.

2. Arduino IDE Basic Composition

Basic composition of Arduino IDE is as below.

1 **Compile button**

2 **Upload button**

3 **Serial monitor button**

4 **Message area**

1. Compile button : Program will be compiled. (Compile result to be shown at area#4)
2. Upload button : Upload to Arduino at the same time of compile (In case of compile error or Arduino & usb connection error occur, it will be shown at area#4.)
3. Serial monitor button : When PC is connected with Arduino via USB, Arduino is able to send message to PC when its operation. If program is written by Serial.write() or Serial.print() functions, user is able to check message on Serial monitor.
4. Message area : Various Error message, complie, upload result will be shown in this area.