Studuino Library

Function Reference

Ver0.9.3

Jan. 27, 2014

This manual is a reference guide for Studuino Library Functions. As the Studuino Programming Environment develops, this manual may be edited or revised.

Index

1. About Studuino Functions Library

The Studuino Function Library is used to talk with your Studuino using Arduino IDE. The library lets you control parts easily because the levels of each part (DC motor, servomotor, etc.) are already configured for you.

2. Function

The explanation of each function will be in the following format:

Function name	**(Function name)**			
Argument	(Type)	(Variable name)	(Value)	(Explanation)
Returns	(Type)	(Explanation)		
Notes				

The following section describes each function.

2.1. Initialize Function

The following section describes functions that are used to initialize ports on your Studuino.

Function name	**InitDCMotorPort**			
Argument	byte	connector	PORT_M1	Connector
			PORT_M2	
Returns	N/A			

Use this function to initialize a DC motor port before using a DC motor.
(Example)
//Use this function to initialize a DC motor port before using the Move, DCMotor, DCMotorPower, or DCMotorControl functions
InitDCMotorPort(PORT_M1); // Initialize a DC motor connected to M1				

Function name	**InitServomotorPort**			
Argument	byte	connector	See #1	Connector
Returns	N/A			

Use this function to initialize a servomotor port before using a servomotor.
(Example)
// Use this function to initialize the servomotor port before using the Servomotor, SyncServomotors, or AsyncServotomors functions.
InitServomotorPort(PORT_D2); // Initialize a servomotor connected to D2

Function name	**InitServomotorPortForLED**			
Argument	byte	connector	PORT_D9	Connector
			PORT_D10	
			PORT_D11	
Returns	N/A			

This function initializes servomotor ports D9, D10, and D11 for use with LEDs. Use this function to initialize a port before trying to control the brightness of an LED on port D9, D10, or D11.
(Example)
// Use this function to initialize the port before using the Gradation function
InitServomotorPortForLED (PORT_D9); // Initialize an LED connected to D9
Gradation(PORT_D9, 128);				

Function name	**InitSensorPort**			
Argument	byte	connector	See #4	Connector
	byte	pid	See #5	Connected parts
Returns	N/A			

Use this function to initialize a sensor/LED/buzzer port before trying to use a sensor, LED, or buzzer.
(Example)
// Use this function to initialize a port before using the Buzzer, BuzzerControl, Melody, LED, or Get* functions
InitSensorPort(PORT_A0, PIDLED); // Initialize an LED connected to port A0				

2.2. DC Motor Control Function

The following section describes DC motor control functions.

Function name	**Move**			
Argument	byte	direct	FORWARD	Forward
			BACKWARD	Reverse
			FORWARD_RIGHT	Right turn (front)
			FORWARD_LEFT	Left turn (front)
			BACKWARD_RIGHT	Right turn (rear)
			BACKWARD_LEFT	Left turn (rear)
			CLOCKWISE	Spin right
			COUNTERCLOCKWISE	Spin left
	byte	pace	0～255	Speed (level)
	ulong	duration	0～2^32-1	Time(msec)
	byte	brake	BRAKE	Brake ON
			COAST	Brake OFF
Returns	N/A			

These functions are used to control motion in a car with two DC motors. The performance of these functions depends on how the DC motors are connected to your Studuino. Build a car using the guide in **A. Connecting DC Motors to Studuino.**

(Example)

Move (FORWARD, 10, 1000, BRAKE); // The car will continue its speed at level 10 for one second then stop

Function name	**DCMotor**			
Argument	byte	connector	PORT_M1	Connector
			PORT_M2	
	byte	rotation	NORMAL	Spin right
			REVERSE	Spin left
	byte	pace	0～255	Speed (level)
	ulong	duration	0～2^32-1	Time(msec)
	byte	brake	BRAKE	Brake ON
			COAST	Brake OFF
Returns	N/A			

This function controls a DC motor.

(Example)

// The DC motor connected to M1 will rotate at a level 10 speed for one second then brake.

DCMotor (PORT_M1, NORMAL, 10, 1000, BRAKE);

Function name	**DCMotorPower**			
Argument	byte	connector	PORT_M1	Connector
			PORT_M2	
	byte	pace	0～255	Speed (level)
Returns	N/A			

This function controls the speed of a DC motor.

(Example)

//The DC motor connected to M1 will rotate at a level 10 speed for one second then stop. Then, is will rotate at 100 for one second then stop.

DCMotorPower(PORT_M1, 10); // M1Sets the speed of the DC motor connected to M1.

DCMotorControl(PORT_M1, CLOCKWISE); // Makes the DC motor connected to M1 move forward

Timer(1000); // Counts one second

DCMotorPower(PORT_M1, 100); // Changes the speed of M1's DC motor

Timer(1000); // Counts one second

DCMotorControl (PORT_M1, BRAKE); // Stops the DC motor connected to M1

Function name	**DCMotorControl**			
Argument	byte	connector	PORT_M1	Connector
			PORT_M2	
	byte	rotation	NORMAL	Spin right
			REVERSE	Spin left
			BRAKE	Brake ON
			COAST	Brake OFF
Returns	N/A			

This function controls the rotation of a DC motor.

(Example)

//The DC motor connected to M1 will rotate at a level 10 speed for one second then stop

DCMotorPower(PORT_M1, 10); // Sets the speed of the DC motor connected to M1.
DCMotorControl(PORT_M1, CLOCKWISE); // Makes the DC motor connected to M1 move forward.
Timer(1000); // Counts one second
DCMotorControl (PORT_M1, BRAKE); // Stops the DC motor connected to M1

2.3. Servomotor Control Function

This following section describes servomotor control functions.

Function name	**Servomotor**			
Argument	byte	connector	See #1	Connector
	byte	degree	0～180	Servomotor angle
Returns	N/A			

This function sets the angle for one servomotor. This function allows you to use another process while the angle is being set.
(Example)
// Set the servomotor connected to D2 at a 90° angle.
Servomotor (PORT_D2, 90);

Function name	**AsyncServomotors**			
Argument	byte[]	connectors	See #1	Connector order
	byte[]	degrees	0～180	Degree of each servomotor
	byte	number	1～8	Number of servomotors
Returns	N/A			

This function sets the angle for multiple servomotors. This function allows you to use another process while the angles are being set.
(Example)
// Set each servomotor connected to D2, D9, and D10 to 90°, 180° , and 45°.
byte myConectors[] = { PORT_D2, PORT_D9, PORT_D10 };
byte myDegrees[] = { 90, 180, 45};
ASyncServomotor (myConectors, myDegrees, 3);

Function	**SyncServomotors**

name				
Argument	byte[]	connector	See #1	Connector order
	byte[]	degree	0～180	Degree of each servomotor
	byte	number	1～8	Number of servomotors
	byte	time	0～255	Movement time (ms) per 1°
Returns	N/A			
This function sets the angle for multiple servomotors. This function will not perform any actions until the angle for the servomotor has been set. Use the time argument to change how fast or slowly the angle is set. However, if the time programmed is less than 3 the servomotor's movement speed will not change due to 3 ms being the top speed per 1° change. (Example) // Set each servomotor connected to D2, D9, and D10 to 90°, 180° , and 45°. byte myConectors[] = { PORT_D2, PORT_D9, PORT_D10 }; byte myDegrees[] = { 90, 180, 45}; SyncServomotor (myConectors, myDegrees, 3, 5);				

2.4. Buzzer Control Function

The following section describes buzzer control functions.

Function name	**Buzzer**			
Argument	byte	connector	See #3	Connector
	word	pitch	See #6	Sound pitch
	ulong	duration	0～2^32-1	Input time (msec)
Returns	N/A			
Plays a designated note for the specified length of time. (Example) Buzzer (PORT_A0, BZR_C4, 1000); // Plays the note "Do" from the buzzer connected to A0 for one second.				

Function name	**BuzzerControl**

Argument	byte	connector	See #3	Connector
	boolean	onoff	ON	Sound output
			OFF	Stops sound
	byte	pitch	See #6	Pitch
Returns	N/A			

When the buzzer is set to ON, the buzzer will play the designated note. When the buzzer is set to OFF, there is no sound (designated pitch value will be nulled).

(Example)

```
// Plays the note "Do" from the buzzer connected to A0 for one second.
BuzzerControl(PORT_A0, ON, BZR_C4);
Timer(1000);
BuzzerControl(PORT_A0, OFF, 0);
```

Function name	**Melody**			
Argument	byte	connector	See #3	Connector
	word[]	pitches	See #6	Pitch
	float[]	beats	0～	Beat
	byte	number	Melody speed (0 - 225)	Number of notes
	byte	tempo	TEMPO60	Tenpo
			TEMPO90	
			TEMPO120	
			TEMPO150	
Returns	N/A			

The buzzer plays a programmed melody.

(Example)

```
// Plays the melody "Do Re Mi Fa Mi Re Do" from the buzzer connected to A0.
word myPitches[] = { BZR_C3, BZR_D3, BZR_E3, BZR_F3,
                     BZR_E3, BZR_D3, BZR_C3 };
float myBeats[] = { 1, 1, 1, 1, 1, 1, 1 };
byte num = 7;    // Number of items
Melody (PORT_A0, myPitches, myBeats, num, TEMPO90);
```

Longer melodies may exceed the amount of SRAM on the Studuino board. When the data space has exceeded on your SRAM, type the keyword PROGMEM and you can save and

use the space in the Flash memory data space. You can view the webpage below for more information by typing in the keyword **PROGMEM**.

http://www.musashinodenpa.com/arduino/ref/index.php?f=0&pos=1824

2.5. LED Control Function

The following section describes LED control functions.

Function name	**LED**			
Argument	byte	connector	See #3	Connector
	boolean	onoff	ON	LED ON/OFF
			OFF	
Returns	N/A			
Turns the LED lights off and on. (Example) LED (PORT_A0, ON); // Turns the LED light connected to A0 on.				

Function name	**Gradation**			
Argument	byte	connector	PORT_D9	Connector
			PORT_D10	
			PORT_D11	
	byte	ratio	0～255	Brightness (becomes brighter when the brightness value is greater)
Returns	N/A			
Use this function to adjust the brightness of LEDs connected to D9, D10, or D11. (Example) Gradation (PORT_D9, 128); // Set the brightness of the LED connected to D9.				

2.6. Input Function

The following section describes Studuino push-button and sensors functions.

Function	**GetPushSwitchValue**

name				
Argument	byte	connector	See #2	Connector
Returns	byte	0: pushed, 1: released		

Checks the condition of the push switch
(Example)
// Check the value of the push switch connected to A0
byte val = GetPushSwitchValue (PORT_A0);				

Function name	**GetTouchSensorValue**			
Argument	byte	connector	See #3	Connector
Returns	byte	0: pushed, 1: released		

Checks the value of the touch sensor
(Example)
// Check the value of the touch sensor connected to A0
byte val = GetTouchSensorValue (PORT_A0);				

Function name	**GetLightSensorValue**			
Argument	byte	connector	See #4	Connector
Returns	int	0～1023		
Checks the value of the light sensor (Example) int val = GetLightSensorValue (PORT_A0); // Check the value of the light sensor connected to A0.				

Function name	**GetSoundSensorValue**			
Argument	byte	connector	See #4	Connector
Returns	int	0～1023		
Checks the value of the sound sensor (Example) int val = GetSoundSensorValue (PORT_A0); // Check the value of the sound sensor connected to A0.				

Function name	GetIRPhotoreflectorValue			
Argument	byte	connector	See #4	Connector
Returns	int	0～1023		

Checks the value of the reflective infrared sensor
(Example)
// Check the value of the reflective infrared sensor connected to A0
int val = GetIRPhotoreflectorValue (PORT_A0);				

Function name	GetAccelerometerValue			
Argument	byte	axis	X_AXIS	Axis you wish to measure
			Y_AXIS	
			Z_AXIS	
Returns	int	0～1023		
Checks the value of the accelerometer. Because the accelerometer uses I2C it is only compatible with ports A4 and A5.. (Example) int val = GetAccelerometerValue (X_AXIS); // Check the tilt of the of the accelerometer's X axis				

2.7. Timer Function

The following section describes the Timer function.

Function name	Timer			
Argument	ulong	time	0～2^32-1	Time (msec)
Returns	N/A			
Counts for the specified time (Example) Timer (1000); // Counts one second				

*** 1 Servomotor Port Settings**

Value	Port

PORT_D2	D2
PORT_D4	D4
PORT_D7	D7
PORT_D8	D8
PORT_D9	D9
PORT_D10	D10
PORT_D11	D11
PORT_D12	D12

*** 2 Push Switch Settings**

Value	Port
PORT_A0	A0
PORT_A1	A1
PORT_A2	A2
PORT_A3	A3

Value	Port
PORT_A0	A0
PORT_A1	A1
PORT_A2	A2
PORT_A3	A3
PORT_A4	A4
PORT_A5	A5

*** 4 Analog Port Settings**

Value	Port
PORT_A0	A0
PORT_A1	A1
PORT_A2	A2
PORT_A3	A3
PORT_A4	A4
PORT_A5	A5
PORT_A6	A6
PORT_A7	A7

*** 3 Digital Port Settings**

*** 5 Part ID**

Value	Part
PIDOPEN	Disconnected
PIDLED	LED
PIDBUZZER	Buzzer
PIDLIGHTSENSOR	Light sensor
PIDSOUNDSENSOR	Sound sensor
PIDIRPHOTOREFLECTOR	Reflective infrared sensor
PIDACCELEROMETER	Accelerometer
PIDTOUCHSENSOR	Touch sensor
PIDPUSHSWITCH	Push switch

*** 6 Pitch Value**

Value	Scale	Hz
BZR_C3	Do	130
BZR_CS3	Do #	139
BZR_D3	Re	147
BZR_DS3	Re #	156
BZR_E3	Me	165
BZR_F3	Fa	175
BZR_FS3	Fa #	185
BZR_G3	So	196
BZR_GS3	So #	208
BZR_A3	La	220
BZR_AS3	La #	233
BZR_B3	Ti	247
BZR_C4	Do	262
BZR_CS4	Do #	277
BZR_D4	Re	294
BZR_DS4	Re #	311
BZR_E4	Mi	330
BZR_F4	Fa	349
BZR_FS4	Fa #	370
BZR_G4	So	392
BZR_GS4	So #	415
BZR_A4	La	440
BZR_AS4	La #	466
BZR_B4	Ti	494

Value	Scale	Hz
BZR_C5	Do	523
BZR_CS5	Do #	554
BZR_D5	Re	587
BZR_DS5	Re #	622
BZR_E5	Mi	659
BZR_F5	Fa	698
BZR_FS5	Fa #	740
BZR_G5	So	784
BZR_GS5	So #	831
BZR_A5	La	880
BZR_AS5	La #	932
BZR_B5	Ti	988
BZR_C6	Do	1047
BZR_CS6	Do #	1109
BZR_D6	Re	1175
BZR_DS6	Re #	1245
BZR_E6	Mi	1319
BZR_F6	Fa	1397
BZR_FS6	Fa #	1480
BZR_G6	So	1568
BZR_GS6	So #	1661
BZR_A6	La	1760
BZR_AS6	La #	1865
BZR_B6	Ti	1976

Value	Scale	Hz
BZR_C7	Do	2093
BZR_CS7	Do #	2217
BZR_D7	Re	2349
BZR_DS7	Re #	2489
BZR_E7	Mi	2637
BZR_F7	Fa	2794
BZR_FS7	Fa #	2960
BZR_G7	So	3136
BZR_GS7	So #	3322
BZR_A7	La	3520
BZR_AS7	La #	3729
BZR_B7	Ti	3951
BZR_C8	Do	4186
BZR_S	No Sound	0

3. Programming

The following section describes items you should keep in mind when programming using Studuino Library functions.

3.1. Arduino language

You must be able to define the following setup function and loop functions in Arduino language. The setup function will only be run once. The processes defined in the loop function will be infinitely repeated.

```
// Used to initialize a program. Only runs once.
void setup() {
   // Use initialize functions to open ports on Studuino that have parts connected to them.
}

// This function is repeated infinitely and is the main process of your program.
void loop() {
}
```

3.2. Studuino Object

When using Studuino Library, it is important to make the Studuino object reflect the image of one Studuino board by using a global variable.

```
//Recognizes the Studuino board. Include only once per program。
Studuino board;
```

3.3. Including a Header File

Include header files when using Studuino Library functions for servomotors and accelerometers. These header files must be included with the Studuino header files.

```
#include <Arduino.h >       // Basic header file
#include <Servo.h>          // Header file for servomotor
#include <Wire.h>           // Header file for accelerometer
#include <MMA8653.h>        // Header file for accelerometer
#include  “Stduino.h”       // Header file for Studuino
```

3.4. Program Examples

The following section contains programming examples for each part.

3.4.1. DC Motor Program Example

Connect the DC motor to M1 and M2 of your Studuino board using **Appx. A.Connecting DC Motors to Studuino** as a reference. Send the following program to your Studuino using Arduino IDE. The DC motor connected to M1 will move forward for a one second interval. The car will move forward for one second then reverse for one second.

```
#include <Arduino.h >        // Basic header file
#include <Servo.h>           // Header file for servomotor
#include <Wire.h>            // Header file for accelerometer
#include <MMA8653.h>         // Header file for accelerometer
#include  "Studuino.h"       // Header file for Studuino

// Recognizes the Studuino board. Include only once per program.
Studuino board;

//  Used to initialize a program. Only runs once.
void setup() {
   // Use initialize functions to open ports on Studuino that have parts connected to them
   board.InitDCMotorPort(PORT_M1);      // Initialize the DC motor connected to M1.
   board.InitDCMotorPort(PORT_M2);      // Initialize the DC motor connected to M2.
}

// This function is repeated infinitely and is the main process of your program.
void loop() {
   board.Move(FORWARD, 254, 1000, BRAKE);         // Moves forward for 1 second and stops
   board.Move(BACKWARD, 254, 1000, BRAKE);        // Moves in reverse for 1 second and stops

  // The DC motor connected to M! will move forward for a one second interval then stop.
   board.DCMotorPower(PORT_M1, 254);              // Sets the rotation speed of the DC motor
                                                                    conencted to M1
   board.DCMotorControl(PORT_M1, NORMAL);         // Starts the DC motor connected to M1 to
                                                                        move forward
   board.Timer(1000);                             // Waits one second
   board.DCMotorControl(PORT_M1, BRAKE);          // Stops DC motor on M1

   for (;;) {}        //Insert a loop so that the process does not start again from the
                                                                                top.
}
```

3.4.2. Servomotor Program Example

Connect servomotors to D10, D11, and D12. Send the following program to your Studuino using Arduino IDE. Initializes all servomotors at 90° after approximately three seconds, servomotors D10, D11, and D12 will be set simultaneously to 90°, 180° until 0° at a low speed. After approximately three seconds, the servomotor in D10 will be set to 180°.

```
#include <Arduino.h >        // Basic header file
#include <Servo.h>           // Header file for servomotor
#include <Wire.h>            // Header file for accelerometer
#include <MMA8653.h>         // Header file for accelerometer
#include  "Studuino.h"       // Header file for Studuino

// Recognizes the Studuino board. Include only once per program.
Studuino board;

// Used to initialize a program. Only runs once.
void setup() {
   //  Use initialize functions to open ports on Studuino that have parts connected to them
   board.InitServomotorPort(PORT_D10);//Initialize the servomotor connected to D10
   board.InitServomotorPort(PORT_D11);//Initialize the servomotor connected to D11
   board.InitServomotorPort(PORT_D12);// Initialize the servomotor connected to D12
}
// This function is repeated infinitely and is the main process of your program.
void loop() {
   //Initialize servomotors at 90°
   byte connector[] = { PORT_D10, PORT_D11, PORT_D12 };
   byte degree[] = { 90, 90, 90 };
   byte number = sizeof(connector) / sizeof(byte); // Number of degree angles set for each
                                                                                    port

   board.AsyncServomotors(connector, degree, number);
   // If necessary, you can input a delay in the servomotor's movement until it reaches its
                                                                                    set angle.
   board.Timer(1000);

   board.Timer(3000);          // Waits three seconds

   // Sets servomotors D10, D11, and D12 at 90° , 180° , and 0°  angles.
   degree[0] = 90;
   degree[1] = 180;
   degree[2] = 0;

   // The servomotor will move until it reaches the set angle after this function has been
                                                                                    input
   board. SyncServomotors(connector, degree, number, 10);

   board.Timer(3000);          // Waits three seconds

   // Sets the degree angle of the servomotor connecting to D10 to 180°
   board.Servomotor(PORT_D10, 180);
   // If necessary, you can input a delay in the servomotor's movement until it reaches its
                                                                                    set degree angle.
   board.Timer(1000);


   for (;;) {}        // Insert a loop so that the process does not start again from the
                                                                                    top.
}
```

3.4.3. Buzzer Program Example

Connect a buzzer to A0 and send the following program to your Studuino using Arduino IDE. After playing "Do" and "So", the buzzer will play the melody for Twinkle Twinkle Little Star.

```
#include <Arduino.h >        // Basic header file
#include <Servo.h>           // Header file for servomotor
#include <Wire.h>            // Header file for accelerometer
#include <MMA8653.h>         // Header file for accelerometer
#include  "Studuino.h"       // Header file for Studuino

//Recognizes the Studuino board. Include only once per program.
Studuino board;


// Used to initialize a program. Only runs once.
void setup() {
   // Use initialize functions to open ports on Studuino that have parts connected to them
   board.InitSensorPort(PORT_A0, PIDBUZZER);// Initialize the buzzer connected to A0
}

// This function is repeated infinitely and is the main process of your program.
void loop() {
   // Stops sound from the buzzer for one second
   board.Buzzer(PORT_A0, BZR_C5, 1000);

   board.Timer(1000);          // Waits one second

   // Stops sound from the buzzer for one second
   board.BuzzerControl(PORT_A0, ON, BZR_G5);
   board.Timer(1000);
   board.BuzzerControl(PORT_A0, OFF, 0);// Last argument will be ignored if the buzzer is
                                                                                    off

   board.Timer(1000);          // Waits one second

   // Plays melody from the buzzer
   word myPitches[] = { BZR_C5, BZR_C5, BZR_G5, BZR_G5, BZR_A5, BZR_A5, BZR_G5 };
   byte number = sizeof(myScales) / sizeof(word); // Number of notes played
   float myBeats[] = { 1, 1, 1, 1, 1, 1, 1 };     // Number of beats
   board.Melody (PORT_A0, myPitches, myBeats, number, TEMPO90);

   for (;;) {}        // Insert a loop so that the process does not start again from the
                                                                                    top.
}
```

3.4.4. LED Program Example

Connect the LED lights to A1 and D9 and send the following program to your Studuino using Arduino IDE. D9 LED will blink slowly after A1 LED blinks three times.

```
#include <Arduino.h >        // Basic header file
#include <Servo.h>           // Header file for servomotor
#include <Wire.h>            // Header file for accelerometer
#include <MMA8653.h>         // Header file for accelerometer
#include  "Studuino.h"       //  Header file for Studuino


// Recognizes the Studuino board.  Include only once per program.
Studuino board;


// Used to initialize a program.  Only runs once.
void setup() {
   // Use initialize functions to open ports on Studuino that have parts connected to them
   board.InitSensorPort(PORT_A1, PIDLED);// Initialize the LED connecting to A0
   board.InitServomotorPortForLED(PORT_D9);//  Initialize the LED connected to D9
}

// This function is repeated infinitely and is the main process of your program.
void loop() {
   //  LED light connected to A1 will blink 3 times
   for (int i = 0;i < 3;i++) {
     board.LED(PORT_A1, ON);   // A1 LED blinks
     board.Timer(1000);        // Wait one second
     board.LED(PORT_A1, OFF);  // A1 LED blinks
     board.Timer(1000);        // Wait one second
   }

   //  LED connected to D9 blinks slowly
   board.Gradation(PORT_D9, 0);
   for (int i = 0;i < 255;i++) {
     board.Gradation(PORT_D9, i);
     board.Timer(100);
   }

   for (;;) {}        // Insert a loop so that the process does not start again from the
                                                                                top.
}
```

3.4.5. Examples of Programs Using Sensors

Connect the touch sensor to A1, sound sensor to A2, reflective infrared sensor to A3, accelerometer to A4/A5, and the light sensor to A6. Send the following program to your Studuino using Arduino IDE. After sending the program, go to Arduino IDE **Menu** > **Tools** > select **Serial Monitor**. Then the serial monitor menu will display each sensors value.

```
#include <Arduino.h >        // Basic header file
#include <Servo.h>           // Header file for servomotor
#include <Wire.h>            // Header file for accelerometer
#include <MMA8653.h>         // Header file for accelerometer
#include  "Studuino.h"       // Header file for Studuino


// Recognizes the Studuino board.  Include only once per program.
```

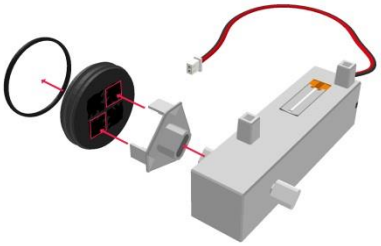
```
Studuino board;

// Used to initialize a program. Only runs once.。
void setup() {
   // Use initialize functions to open ports on Studuino that have parts connected to them
   board.InitSensorPort(PORT_A0, PIDPUSHSWITCH);
   board.InitSensorPort(PORT_A1, PIDTOUCHSENSOR);
   board.InitSensorPort(PORT_A2, PIDSOUNDSENSOR);
   board.InitSensorPort(PORT_A3, PIDIRPHOTOREFLECTOR);
   board.InitSensorPort(PORT_A4, PIDACCELEROMETER);
   board.InitSensorPort(PORT_A5, PIDACCELEROMETER);
   board.InitSensorPort(PORT_A6, PIDLIGHTSENSOR);

   // Initialize the serial output
   Serial.begin(9600);
}

// This function is repeated infinitely and is the main process of your program.
void loop() {
   // The serial monitor will show the light sensor value in units of 100 msec
   for (;;) {
     byte pVal = board.GetPushSwitchValue(PORT_A0);
     byte tVal = board.GetTouchSensorValue(PORT_A1);
     int  sVal = board.GetSoundSensorValue(PORT_A2);
     int  iVal = board.GetIRPhotoreflectorValue(PORT_A3);
     int  xVal = board.GetAccelerometerValue(X_AXIS);
     int  yVal = board.GetAccelerometerValue(Y_AXIS);
     int  zVal = board.GetAccelerometerValue(Z_AXIS);
     int  lVal = board.GetLightSensorValue(PORT_A6);
     Serial.print("button:");  Serial.print(pVal);        Serial.print("¥t");
     Serial.print("touch:");   Serial.print(tVal);        Serial.print("¥t");
     Serial.print("sound:");   Serial.print(sVal);        Serial.print("¥t");
     Serial.print("ir:");      Serial.print(iVal);        Serial.print("¥t");
     Serial.print("x:");       Serial.print(xVal);        Serial.print("¥t");
     Serial.print("y:");       Serial.print(yVal);        Serial.print("¥t");
     Serial.print("z:");       Serial.print(zVal);        Serial.print("¥t");
     Serial.print("light:");   Serial.print(lVal);        Serial.println();
     board.Timer(100);
   }
}
```

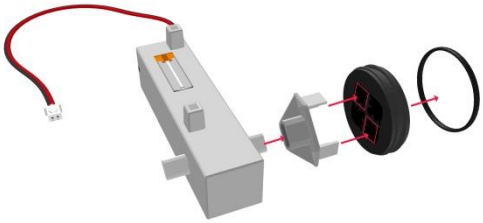
Appendix A. Connecting the Studuino board and DC Motor

Assemble the car as follows.

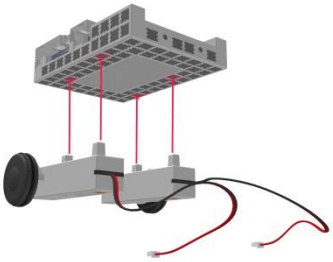
（1）Attach wheels to the DC motors.

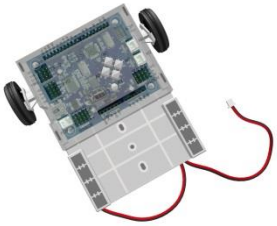
★ Make symmetrical pair.



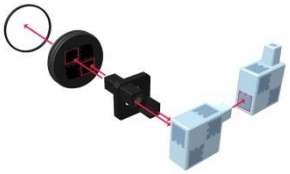


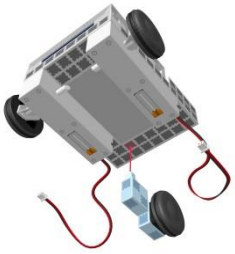
(2) Attach both DC motors to the bottom of the base mount.



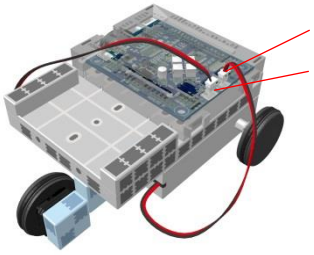


（3）Use blocks to make a rear wheel





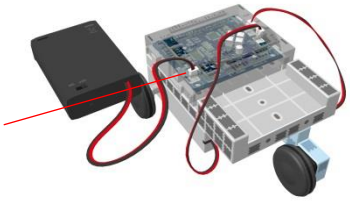
（4）Connect the DC motors and battery box to the Studuino board.



M1：Right-side DC motor

M2：Left-side DC motor

POWER：Battery box



（5）Secure the battery box to the base mount.