Studuino 功能庫

功能參考

Published 2014/11/01
Revised 2018/02/01

版本歷史

日期	內容
2014/11/01	首個版本
2017/01/16	就新的 Studuino 網站作更新
2017/08/16	在程式範例中,將 Stduino.h 更改為 Studuino.h
2018/02/01	修訂文本
2022/06/07	中文版本

目錄

1. 入門

本手冊涵蓋了 Studuino 在 Arduino IDE 中使用到的 Studuino 功能庫及 ArtecRobo 電子部件所需的功能，包括直流馬達及伺服馬達，以供編程之用。

手冊內容將跟據軟件更新而作出改動。

2. 功能

功能描述將採用以下格式：

功能名稱:	**(功能名稱)**			
參數	(類型)	(變數名稱)	(值)	(描述)
回傳值	(類型)	(描述)		
備註				

2.1. 初始化功能

功能名稱:	**SetDCMotorCalibration**			
參數	byte[2]	速度	0 ~ 100	偏移值
回傳值	無			

您只需要使用此功能來調整直流馬達的速度。

此功能用作調整直流馬達的速度。更改 **rate [0]** 的參數值以控制 M1 直流馬達的速度，更改 **rate [1]** 的參數值以控制 M2 直流馬達的速度。

(例子)

```
// 將 M1 直流馬達的最高速度設定為 80%
// 將 M2 直流馬達的最高速度設定為 100%
byte calib[] = { 80, 100 };
SetDCMotorCalibration(calib); // 直流馬達速度校准設定
```

功能名稱:	**SetServomotorCalibration**			
參數	char[8]	偏移角度	-15 ~ 15	偏移值
回傳值	無			

此功能僅用於調整伺服馬達的初始角度。

此功能調整伺服馬達的角度。 更改偏移值 **[0] ~ [7]** 中的參數值以指定 D2 ~ D12 上相應伺服馬達的角度。

(例子)

//指定連接到 D9 ~ D12 的伺服馬達

byte calib[] = { 0, 0, 0, 0, -6, 0, 12, 3 }; // 角度: D9(-6°), D10(0°), D11(12°), D12(3°)

SetServomotorCalibration(calib); //伺服馬達速度校准設定

功能名稱:	**InitDCMotorPort**			
參數	byte	端口	PORT_M1 PORT_M2	端口
回傳值	無			

此功能初始化一個直流馬達端口。 在使用直流馬達前，請使用此功能初始化端口。

(例子)

// 使用此功能初始化端口後,才可使用功能 Move, DCMotor, DCMotorPower 或 DCMotorControl functions

InitDCMotorPort(PORT_M1); //初始化端口 M1 以用於直流馬達

功能名稱:	**InitServomotorPort**			
參數	byte	端口	見 CH2.9 * 1	端口
回傳值	無			

此功能初始化一個伺服馬達端口。 在使用伺服馬達前，請使用此功能初始化端口。

(例子)

// 使用此功能初始化端口後,才可使用功能 Servomotor, SyncServomotors 或 AsyncServomotors

InitServomotorPortPORT_D2); // 初始化端口 D2 以用於伺服馬達

功能名稱:	InitServomotorPortForLED			
參數	byte	端口	PORT_D9	端口
			PORT_D10	
			PORT_D11	
回傳值	無			

初始化伺服馬達端口 D9、D10 或 D11 以使用 LED。在調整 D9、D10 或 D11 上的 LED 亮度之前，使用此功能初始化端口。

(例子)
//使用此功能初始化端口後,才可使用功能 Gradation
InitServomotorPortForLEDPORT_D9); // 初始化端口 D9 以與 LED 一起使用
Gradation(PORT_D9, 128);

功能名稱:	**InitSensorPort**			
參數	byte	端口	見 CH2.9 * 4	端口
	byte	部件 ID	見 CH2.9 * 5	部件
回傳值	無			

此功能初始化傳感器、蜂鳴器或 LED 的端口。 在將端口與傳感器、蜂鳴器或 LED 一起使用之前，使用此功能初始化端口。

(例子)
// 使用此功能初始化端口後,才可使用功能 Buzzer, BuzzerControl, Melody, LED 或 Get*
InitSensorPort(PORT_A0, PIDLED); //初始化端口 A0 以與 LED 一起使用

功能名稱:	**InitSensorPort**			
參數	byte	端口 1	見 CH2.9 * 4	端口
	byte	端口 2	見 CH2.9 * 4	端口
	byte	部件 ID	見 CH2.9 * 5	部件
回傳值	無			

此功能初始化超聲波傳感器的端口。 在將端口與超聲波傳感器一起使用之前，使用此功能初始化端口。

(例子)
InitSensorPort(PORT_A0, PORT_A1, PIDULTRASONICSENSOR); // 初始化端口 A0 和 A1 以用於超聲波傳感器

功能名稱:	**InitI2CPort**			
參數	byte	部件 ID	見 CH2.9 * 5	部件 (僅限 I2C 設備)
回傳值	無			

此功能初始化 I2C 端口（A4、A5）。 在將端口與 I2C 設備一起使用之前，使用此功能初始化端口。

(例子)
InitI2CPort(PIDACCELEROMETER); //初始化 I2C 端口以與速度感應器一起使用

功能名稱:	InitBluetooth
參數	無
回傳值	無

通過藍牙打開串行連接。串口初始傳輸速率: 9600
此功能支援以下 2 藍牙個模組:
RBT-001 Bluetooth Module (product #86873)
Bluetooth Module for Robots (product #86876）

(例子)
InitBluetooth(); // 初始化藍牙

2.2. 直流馬達功能

本節介紹用於控制直流馬達的功能。

<table>
<tr><td>功能名稱:</td><td colspan="4">Move</td></tr>
<tr><td rowspan="12">參數</td><td rowspan="8">byte</td><td rowspan="8">方向</td><td>FORWARD</td><td>直走</td></tr>
<tr><td>BACKWARD</td><td>倒退</td></tr>
<tr><td>FORWARD_RIGHT</td><td>右轉（前）</td></tr>
<tr><td>FORWARD_LEFT</td><td>左轉（前）</td></tr>
<tr><td>BACKWARD_RIGHT</td><td>右轉（後）</td></tr>
<tr><td>BACKWARD_LEFT</td><td>左轉 (後)</td></tr>
<tr><td>CLOCKWISE</td><td>順時針轉動</td></tr>
<tr><td>COUNTERCLOCKWISE</td><td>逆時針轉動</td></tr>
<tr><td>byte</td><td>速度</td><td>0 ~ 255</td><td>速度</td></tr>
<tr><td>ulong</td><td>時間</td><td>0 ~ 2^32-1</td><td>時間（毫秒）</td></tr>
<tr><td rowspan="2">byte</td><td rowspan="2">剎車</td><td>BRAKE</td><td>使用剎車</td></tr>
<tr><td>COAST</td><td>解除剎車</td></tr>
<tr><td>回傳值</td><td colspan="4">無</td></tr>
<tr><td colspan="5">此功能用於控制使用汽車運動的兩個直流馬達。 直流馬達需要以特定方式連接到板上才能使用。詳情請見”將直流馬達連接到 Studuino “(p.43)

(例子)
Move (FORWARD, 10, 1000, BRAKE); // 汽車以 10 速向前行駛 1 秒後停止</td></tr>
</table>

功能名稱:	DCMotor			
參數	byte	端口	PORT_M1	端口
			PORT_M2	
	byte	轉向	NORMAL	向前
			REVERSE	向後
	byte	速度	0 ~ 255	速度
	ulong	時間	0 ~ 2^32-1	時間（毫秒）
	byte	剎車	BRAKE	使用剎車
			COAST	解除剎車
回傳值	無			

此功能控制單個直流馬達。

(例子)
// 使 M1 上的直流馬達以 10 的速度旋轉一秒鐘並停止
DCMotor (PORT_M1, NORMAL, 10, 1000, BRAKE);

功能名稱:	DCMotorPower			
參數	byte	端口	PORT_M1	端口
			PORT_M2	
	byte	速度	0 ~ 255	速度
回傳值	無			

此功能控制單個直流馬達的速度。

(例子)
// 使 M1 直流馬達以 10 速旋轉一秒，以 100 速旋轉一秒，然後停止
DCMotorPower(PORT_M1, 10); // 設定 M1 直流馬達的速度
DCMotorControl(PORT_M1, CLOCKWISE); // 設定 M1 直流馬達以順時針方向轉動
Timer(1000); // 計時一秒
DCMotorPower(PORT_M1, 100); // 更改 M1 直流馬達的速度
Timer(1000); // 計時一秒
DCMotorControl(PORT_M1, BRAKE); // 剎停 M1 直流馬達

功能名稱:	DCMotorControl			
參數	byte	端口	PORT_M1	端口
			PORT_M2	
	byte	轉向	NORMAL	向前
			REVERSE	向後
			BRAKE	使用剎車
			COAST	解除剎車
回傳值	無			

此功能控制直流馬達的轉向。

(例子)
//使 M1 直流馬達以 10 速旋轉一秒，然後停止
DCMotorPower(PORT_M1, 10); // 設定 M1 直流馬達的速度
DCMotorControl(PORT_M1, CLOCKWISE); // 設定 M1 直流馬達以順時針方向轉動

```
Timer(1000); // 計時一秒
DCMotorControl(PORT_M1, BRAKE); // 剎停 M1 直流馬達
```

2.3. 伺服馬達功能

本節介紹用於控制伺服馬達的功能。

功能名稱:	**Servomotor**			
參數	byte	端口	見 CH2.9 * 1	端口
	byte	角度	0 ~ 180	伺服馬達角度
回傳值	無			

設定一個伺服馬達角度。程式中的下一個過程將會使伺服電機轉動。

(例子)

```
//將 D2 上的伺服馬達設定為 90 度
Servomotor (PORT_D2, 90);
```

功能名稱:	**AsyncServomotors**			
參數	byte[]	端口	見 CH2.9 * 1	端口排列
	byte[]	角度	0 ~ 180	每個伺服馬達的角度
	byte	數量	1 ~ 8	伺服馬達數量
回傳值	無			

設定多個伺服馬達角度。 程式中的下一個過程將會使伺服電機轉動。

(例子)

```
//將 D2、D9 和 D10 上的伺服馬達設定為 90、180 和 45 度
byte myConnectors[] = { PORT_D2, PORT_D9, PORT_D10 };
byte myDegrees[] = { 90, 180, 45};
ASyncServomotor (myConectors, myDegrees, 3);
```

功能名稱:	SyncServomotors			
參數	byte[]	端口	見 CH2.9 * 1	端口排列
	byte[]	角度	0 ~ 180	每個伺服馬達的角度
	byte	數量	1 ~ 8	伺服馬達數量
	byte	時間	3 ~ 255	每度轉動時間 (毫秒)
回傳值	無			

設定多個伺服馬達角度。程式將等待所有伺服馬達旋轉到指定的角度，才會運行其他程式。時間參數價愈大，則伺服馬達旋轉愈慢。請注意，每度旋轉的最大速度為 3 毫秒，若時間參數少於 3，伺服馬達則只會以每度 3 毫秒旋轉。

(例子)

```
//將 D2、D9 和 D10 上的伺服馬達設定為 90、180 和 45 度
byte myConectors[] = { PORT_D2, PORT_D9, PORT_D10 };
byte myDegrees[] = { 90, 180, 45};
SyncServomotor (myConectors, myDegrees, 3, 5);
```

2.4. 蜂鳴器功能

本節介紹用於控制蜂鳴器的功能。

功能名稱:	**Buzzer**			
參數	byte	端口	見 CH2.9 * 3	端口
	word	音調	見 CH2.9 * 6	音符
	ulong	時間	0 ~ 2^32-1	持續時間(毫秒)
回傳值	無			
在指定的時間段內播放蜂鳴器中的音符。 (例子) Buzzer (PORT_A0, BZR_C4, 1000); //**使用蜂鳴器** A0 **播放音符**"Do"**一秒鐘**				

功能名稱:	**BuzzerControl**			
參數	byte	端口	見 CH2.9 * 3	端口
	boolean	開關	ON	播放聲音
			OFF	停止播放聲音
	byte	音調	見 CH2.9 * 6	音符
回傳值	無			

將參數 onoff 設定為 ON 將使蜂鳴器以指定的音高播放一個音符。 將其設定為 OFF 將忽略原有參數並停止蜂鳴器。
(例子)
// 使用蜂鳴器 A0 播放音符"Do"一秒鐘
BuzzerControl(PORT_A0, ON, BZR_C4);
Timer(1000);
BuzzerControl(PORT_A0, OFF, 0);				

功能名稱:	Melody			
參數	byte	端口	見 CH2.9 * 3	端口
	word[]	音調	見 CH2.9 * 6	音符
	float[]	節拍	0 ~	節拍
	byte	數量	Melody number	音符數量
	byte	節奏	TEMPO60	節奏
			TEMPO90	
			TEMPO120	
			TEMPO150	
回傳值	無			

使用蜂鳴器播放旋律。

(例子)

```
//使用蜂鳴器 A0 播放 Do、Re、Mi、Fa、Mi、Re、Do
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;   //音符數量
Melody (PORT_A0, myPitches, myBeats, num, TEMPO90);
```

2.5. LED 功能

本節介紹用於控制 LED 的功能。

功能名稱:	LED			
參數	byte	端口	見 CH2.9 * 3	端口
	boolean	開關	ON	LED 開關
			OFF	
回傳值	無			
打開或關閉 LED。 (例子) LED (PORT_A0, ON); //打開 A0 上的 LED				

功能名稱:	**Gradation**			
參數	byte	端口	PORT_D9	端口
			PORT_D10	
			PORT_D11	
	byte	比率	0 ~ 255	亮度（數值越高越亮）
回傳值	無			
調整連接到端口 D9、D10 或 D11 的 LED 的亮度。 (例子) Gradation (PORT_D9, 128); // 設定 D9 上 LED 的亮度				

2.6. 輸入功能

本節介紹用於按鈕和感應器的功能。

功能名稱:	**GetPushSwitchValue**			
參數	byte	端口	見 CH2.9 * 2	端口
回傳值	byte	0：按下，1：釋放		

取得按鈕的狀態。
(例子)
// 獲取 A0 按鈕的值
byte val = GetPushSwitchValue (PORT_A0);				

功能名稱:	**GetTouchSensorValue**			
參數	byte	端口	見 CH2.9 * 3	端口
回傳值	byte	0：按下，1：釋放		

取得接觸式感應器的狀態。
(例子)
//獲取 A0 接觸式感應器的值
byte val = GetTouchSensorValue (PORT_A0);				

功能名稱:	**GetLightSensorValue**			
參數	byte	端口	見 CH2.9 * 4	端口
回傳值	int	0 ~ 1023		
取得光感應器的狀態。 (例子) int val = GetLightSensorValue (PORT_A0); // 獲取 A0 光感應器的值				

功能名稱:	**GetSoundSensorValue**			
參數	byte	端口	見 CH2.9 * 4	端口
回傳值	int	0 ~ 1023		
取得聲音感應器的狀態。 (例子) int val = GetSoundSensorValue (PORT_A0); //獲取 A0 聲音感應器的值				

功能名稱:	**GetIRPhotoreflectorValue**			
參數	byte	端口	見 CH2.9 * 4	端口
回傳值	int	0 ~ 1023		

取得紅外光反射器的狀態。
(例子)
//獲取 A0 紅外光反射器的值
int val = GetIRPhotoreflectorValue (PORT_A0);

功能名稱:	**GetAccelerometerValue**			
參數	byte	軸	X_AXIS	加速度方向
			Y_AXIS	
			Z_AXIS	
回傳值	int	-128 ~ 127		

取得加速度感應器的值。加速度感應器只能與 A4/A5 上的 I2C 端口一起使用。

(例子)
int val = GetAccelerometerValue (X_AXIS); // 沿 X 軸取得加速度感應器傾斜

功能名稱:	**GetTemperatureSensorValue**			
參數	byte	端口	見 CH2.9 * 5	端口
回傳值	int	0 ~ 1023		

取得溫度感應器的值。

(例子)
int val = GetTemperatureSensorValue(PORT_A0); //獲取 A0 溫度感應器的值
double temperature = ((((val / 1024.0) * 3.3) - 0.5) / 0.01); //將溫度轉換為攝氏度

功能名稱:	**GetUltrasonicSensorValue**			
參數	byte	引發器	見* 4	端口
	byte	接收器	見* 4	端口
回傳值	unsigned long		超聲波來回所需的時間（以微秒為單位）	

取得超聲波感應器的值。回傳值記錄由 triggerPin 發射的超聲波被 echoPin 拾取所需的時間（以微秒為單位）

(例子)
unsigned long val = GetUltrasonicSensorValue(PORT_A0, PORT_A1); //取得超聲波感應器的值
double dist = val / 58.0; // 將距離轉換為厘米
58 = 29[us/cm] * 2：將聲音傳播一厘米所需的時間（以微秒為單位）乘以 2（往返一次）

* 4: p.29

功能名稱:	**GetGyroscopeValue**			
參數	byte	軸	X_AXIS	加速度方向
			Y_AXIS	
			Z_AXIS	
			GX_AXIS	角速度方向
			GY_AXIS	

			GZ_AXIS	
回傳值	int	-32768 ~ 32767		
取得陀螺儀的加速度和角速度。 陀螺儀只能與 A4/A5 上的 I2C 端口一起使用。 (例子) int val = GetGyroscopeValue(GX_AXIS); // 取得沿陀螺儀 X 軸的角速度				
功能名稱:	**GetIRReceiverValue**			
參數	無			
回傳值	unsigned long		紅外線訊號值	
取得紅外線接收器的值。 (例子) unsigned long val = GetIRReceiverValue(); //取得紅外線接收器的值。				

功能名稱:	**DisableIRReceiver**
參數	無
回傳值	無

停用紅外線接收器。 在使用 M1 上的直流馬達或同一程序中的蜂鳴器之前，您需要使用此功能停用紅外線接收器。
(例子)
board.DisableIRReceiver(); // 停用紅外線接收器
board.Move(BACKWARD, DCMPWR(10), 500, COAST); //使用直流馬達
board.EnableIRReceiver(); // 啟用紅外線接收器	

功能名稱:	EnableIRReceiver
參數	無
回傳值	無

停用紅外線接收器。 在使用 M1 上的直流馬達或同一程序中的蜂鳴器之前，您需要使用此功能停用紅外線接收器。
(例子)
board.DisableIRReceiver(); // 停用紅外線接收器
board.Move(BACKWARD, DCMPWR(10), 500, COAST); //使用直流馬達
board.EnableIRReceiver(); // 啟用紅外線接收器	

功能名稱:	**GetColorSensorValue**			
參數	byte	軸	VALUE_RED	被測顏色的組成
			VALUE_GREEN	
			VALUE_BLUE	
			VALUE_CLEAR	
回傳值	Unsigned int		被測顏色的組成	
取得顏色感應器的值。 通過使用紅色、綠色、藍色和透明濾鏡（無濾鏡）返回檢測到的光的顏色成分。 (例子) unsigned int sv = GetColorSensorValue(VALUE_RED); // 取得顏色感應器檢測到的紅色成分				

功能名稱:	**GetColorSensorXY**		
參數	double*	x	用於存儲顏色坐標的 x 值
	double*	y	用於存儲顏色坐標的 y 值
回傳值	無		

取得顏色感應器檢測到的顏色成分轉換為顏色坐標的結果。 這些結果存儲為 x 和 y 值。
(例子)
double x, y; //設變數存儲數值
GetColorSensorXY(&x, &y); //取得顏色坐標並將它們存儲到 x 和 y			

功能名稱:	**GetColorCode**		
參數	無		
回傳值	byte	COLOR_UNDEF	無法定義的顏色
		COLOR_RED	紅色的
		COLOR_GREEN	綠色的
		COLOR_BLUE	藍色的
		COLOR_WHITE	白色的
		COLOR_YELLOW	黃色
		COLOR_BROWN	棕色的
		COLOR_BLACK	黑色的

確定 Artec Block 是紅色、綠色、藍色、白色、黃色、棕色還是黑色，並返回相應的值。

(例子)
int sv = board.GetColorCode(); // 回傳 Artec Block 的顏色值

功能名稱:	**UpdateBluetooth**	
參數	無	
回傳值	boolean	數據接收狀態

嘗試從藍牙控制器應用程序接收數據，如果接收到數據則返回 TRUE，否則返回 FALSE。

(例子)
```
board.UpdateBluetooth();
boolean fID1 = board.GetBTCommandIDState(BT_ID_01); //檢查應用程序中分配給 ID01 的按鈕是否被按下
if(fID1 == true) {
  // 如果按下按鈕
}
```

功能名稱:	**GetBTCommandIDState**		
參數	byte	id	BT_ID_01 ~ BT_ID_10, BT_ID_ACC
回傳值	boolean	檢查是否啟用了 Bluetooth ID 和加速度感應器	

從 UpdateBluetooth 的值中檢索指定命令 ID 的狀態。

目標	ID	回傳值
屏幕按鈕	BT_ID_01 ~ BT_ID_10	TRUE: 開 FALSE: 關
加速度感應器	BT_ID_ACC	TRUE: 啟用 FALSE: 停用

(例子)

```
board.UpdateBluetooth();
boolean fID1 = board.GetBTCommandIDState(BT_ID_01); //檢查應用程序中分配給 ID01 的按鈕是否被按下
if(fID1 == true) {
   //如果按下按鈕
}
```

功能名稱:	**GetBTAccelValue**			
參數	byte	軸	X_AXIS	加速度方向
			Y_AXIS	
			Z_AXIS	
回傳值	無			

在使用 UpdateBluetooth 接收數據後取得加速度感應器的值。

(例子)

```
board.UpdateBluetooth();
boolean fAcc = board.GetBTCommandIDState(BT_ID_ACC); //確定是否在應用程序內部啟用了加速度感應器
int sv = board.GetBTAccelValue(X_AXIS); // 取得加速度感應器的 X 加速度
if(fAcc & (sv > 0)) {
   // 加加速度感應器啟用且 X 加速度為 True 時運行
}
```

2.7. 定時器功能

本節介紹等待指定時間的功能

功能名稱:	**Timer**			
參數	unsigned long	時間	0 ~ 2^32-1	時間(毫秒)
回傳值	無			
使進程停止指定的時間。 (例子) Timer(1000); // 閒置一秒				

2.8. Studuino mini 功能

以下功能只能與 Studuino mini 一起使用。

功能名稱:	InitClock
參數	無
回傳值	無
初始化 LCD 時鐘的端口。	

功能名稱:	**setTime**			
參數	byte	小時	0 ~ 23	小時
	byte	分鐘	0 ~ 59	分鐘
回傳值	無			
設置 LCD 時鐘的時間。 (例子) board.setTime(9, 0);				

功能名稱:	**setDate**			
參數	unsigned int	年	2000 ~ 2040	年
	byte	月	1 ~ 12	月
	byte	日	1 ~ 31	日
回傳值	無			
設置 LCD 時鐘的日期。 (例子) board.setDate(2016, 4, 1);				

功能名稱:	setAlarm			
參數	byte	小時	0 ~ 23	小時
	byte	分鐘	0 ~ 59	分鐘
回傳值	無			
為 LCD 時鐘設置鬧鐘。 (例子) board.setAlarm(9, 0);				

功能名稱:	**setBackLight**			
參數	byte	紅色	0 ~ 15	16 級從 0 [暗] 到 15 [亮]
	byte	綠色	0 ~ 15	16 級從 0 [暗] 到 15 [亮]
	byte	藍色	0 ~ 15	16 級從 0 [暗] 到 15 [亮]
回傳值	無			

更改 LCD 時鐘的背光顏色。

(例子)
board.setBackLight(15, 10, 5); // 將紅色設置為 15，將綠色設置為 10，將藍色設置為 5

功能名稱:	**backLight**			
參數	boolean	開/關	ON	打開背光
			OFF	關閉背光
回傳值	無			

打開或關閉 LCD 時鐘的背光。
(例子)
board.setBackLight(15, 10, 5); // 將紅色設置為 15，將綠色設置為 10，將藍色設置為 5
board.backLight(ON); //打開背光
board.Timer(1000); // 閒置一秒
clock.backLight(OFF); //關閉背光

★時鐘將使用 setBackLight 中最後指定的顏色打開。

功能名稱:	**clockBuzzer**			
參數	word	音調	見 CH2.9 * 6	音符
	unsigned long	時間		持續時間(毫秒)
回傳值	無			

使用 LCD 時鐘的蜂鳴器播放音樂。
(例子)
board.clockBuzzer(BZR_CS4, 2000);

功能名稱:	**GetHour**	
參數	無	
回傳值	int	小時
從 LCD 時鐘中檢索小時。		

功能名稱:	**GetMinute**	
參數	無	
回傳值	int	分鐘
從 LCD 時鐘中檢索分鐘。		

功能名稱:	**GetYear**	
參數	無	
回傳值	int	年份
從 LCD 時鐘中檢索年份。		

功能名稱:	**GetMonth**	
參數	無	
回傳值	int	月份
從 LCD 時鐘中檢索月份。		

功能名稱:	**GetDay**	
參數	無	
回傳值	int	日
從 LCD 時鐘中檢索日。		

功能名稱:	**GetTemperature**	
參數	無	
回傳值	float	溫度 (℃)
從 LCD 時鐘中檢索溫度。		

功能名稱:	**GetAlarmHour**	
參數	無	
回傳值	int	鬧鐘時間(小時)
從 LCD 時鐘中檢索鬧鐘時間(小時)。		

功能名稱:	**GetAlarmMinute**	
參數	無	
回傳值	int	鬧鐘時間(分鐘)
從 LCD 時鐘中檢索鬧鐘時間(分鐘)。		

功能名稱:	**isAlarmTime**	
參數	無	
回傳值	boolean	檢查當前時間是否與鬧鐘時間一致。

回傳 LCD 時鐘的時間是否與鬧鐘時間匹配。 此功能回傳的值與 LCD 時鐘的板載警報開關和 STOP 按鈕相關聯。

	TRUE	FALSE
音調	見 * 3	・總是 False
開啟	・當前時間與鬧鐘時間一致	・當前時間與鬧鐘時間不符 ・當 TRUE 時,按下 STOP 按鈕
休眠	・當前時間與鬧鐘時間一致 ・當前時間等於鬧鐘時間+5 的倍數	・當前時間與鬧鐘時間不符 ・當 TRUE 時,按下 STOP 按鈕

(例子)

```
for(;;) {
   if(board.isAlarmTime()) {
      board.clockBuzzer(BZR_CS4, 2000);
   }
}
```

功能名稱:	**sleep**
參數	無
回傳值	無
SLEEP_MODE_PWR_SAVE 將激活時鐘的省電模式。	

功能名稱:	**GetOnboardLightSensor**	
參數	無	
回傳值	int	0 ~ 1023
檢索板載光感應器的值。		

功能名稱:	**GetBatteryVoltage**	
參數	無	
回傳值	float	電壓[伏特]
檢索連接到電路板的電池的電壓。		

2.9. 常數

* 1 伺服馬達端口

值	端口
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 按鈕端口

值	端口
PORT_A0	A0
PORT_A1	A1
PORT_A2	A2
PORT_A3	A3

*3 數字端口

值	端口
PORT_A0	A0
PORT_A1	A1
PORT_A2	A2
PORT_A3	A3
PORT_A4	A4
PORT_A5	A5

*4 模似端口值

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

*5 部件 **ID**

值	部件
PIDOPEN	斷開連接
PIDLED	LED
PIDBUZZER	蜂鳴器
PIDLIGHTSENSOR	光感應器
PIDSOUNDSENSOR	聲音感應器
PIDIRPHOTOREFLECTOR	紅外光反射器
PIDACCELEROMETER	加速度計
PIDTOUCHSENSOR	觸摸感應器
PIDPUSHSWITCH	按鈕
PIDIRRECEIVER	紅外接收器
PIDGYROSCOPE	陀螺儀 (*)
PIDTEMPERATURESENSOR	溫度感應器
PIDULTRASONICSENSOR	超聲波感應器
PIDCOLORSENSOR	顏色感應器 (*)

(*): I2C device

*6 音符值

值	音符	頻率 (Hz)
BZR_C3	Do	130
BZR_CS3	Do #	139
BZR_D3	Re	147
BZR_DS3	Re #	156
BZR_E3	Mi	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

值	音符	頻率 (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

值	音符	頻率 (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	Silence	0

3. 編程

在使用 Studuino 功能庫編程時，您必須牢記這些部分中的要點。

3.1. Arduino 語言

使用 Arduino 語言時，用戶必須自己定義設置和循環功能。 setup 功能只在程序啟動時調用一次。 循環功能用於無限次重複定義的過程。

```
// 在程序開始時調用一次。 主要用於初始化。
void setup() {
   // 使用初始化功能初始化連接部件的 Studuino 端口
}

// 這個功能運行一個無限循環。 如：主要流程。
void loop() {
}
```

3.2. Studuino 物件

為了使用 Studuino 庫，您必須通過將 Studuino 對象放在全局變量中來創建 Studuino 板的圖像。

```
// Studuino 板圖像。 每個程序只製作一個。
Studuino board;       ★ Studuino
StduinoMini board;      ★ Studuino mini
```

3.3. 包括標頭檔

這些功能數供您的伺服電機、加速度計、陀螺儀、IR 接收器和顏色傳感器使用。 除了 Studuino 的標頭檔之外，您還必須包含這些頭文件。

```
#include <Arduino.h >// 基本標頭檔
#include <Servo.h>              // 伺服馬達標頭檔
#include <Wire.h>               // I2C 設備標頭檔
#include <MMA8653.h> // 加速度感應器標頭檔
#include <MPU6050.h>        // 陀螺儀標頭檔
#include <IRremoteForStduino.h>       // 紅外接收器標頭檔
#include <ColorSensor.h>    // 顏色感應器標頭檔
#include "Studuino.h"// Studuino 標頭檔
```

3.4. 編程示例

以下部分包含每個部件的編程示例。

3.4.1. 直流馬達

將直流馬達連接到 Studuino 上的 M1 和 M2，並使用 Arduino IDE 加載以下程序。汽車將前進一秒鐘，然後倒退一秒鐘。

```
#include <Arduino.h >// 基本標頭檔
#include <Servo.h>              // 伺服馬達標頭檔
#include <Wire.h>               // I2C 設備標頭檔
#include <MMA8653.h> // 加速度感應器標頭檔
#include <MPU6050.h>        // 陀螺儀標頭檔
#include <IRremoteForStuduino.h>        // 紅外接收器標頭檔
#include <ColorSensor.h>    // 顏色感應器標頭檔
#include "Studuino.h"// Studuino 標頭檔

// Studuino 板圖像。 每個程序只製作一個。
Studuino board;

// 在程序開始時調用一次。 主要用於初始化。
void setup() {
   // 使用初始化功能初始化連接部件的 Studuino 端口
   board.InitDCMotorPort(PORT_M1);      // 初始化端口 M1 以用於直流馬達
   board.InitDCMotorPort(PORT_M2);      // 初始化端口 M2 以用於直流馬達
}

// 這個功能運行一個無限循環。 如：主要流程。
void loop() {
   board.Move(FORWARD, 254, 1000, BRAKE);          // 向前行駛一秒鐘然後停下
   board.Move(FORWARD, 254, 1000, BRAKE);          // 倒車一秒鐘然後停止

   // 順時針旋轉直流馬達 M1 1 秒後停止
   board.DCMotorPower(PORT_M1, 254);               // 設置直流馬達 M1 的速度
   board.DCMotorControl(PORT_M1, NORMAL);          // 開始順時針旋轉直流馬達 M1
   board.Timer(1000);                              // 等待一秒鐘
   board.DCMotorControl(PORT_M1, BRAKE);           // 停止直流馬達 M1

   for (;;) {}         // 無限循環以防止程序從頂部重新開始
}
```

3.4.2. 伺服馬達

將伺服馬達連接到 Studuino 上的 D10、D11 和 D12，並使用 Arduino IDE 加載以下程序。 所有伺服馬達將初始化為 90°。 程序將等待三秒鐘，然後將所有三個伺服馬達同時轉動到 90°、180° 和 0°。 然後它將再等待三秒鐘，然後將伺服馬達 D10 旋轉到 180°。

```
#include <Arduino.h >// 基本標頭檔
#include <Servo.h>              // 伺服馬達標頭檔
#include <Wire.h>               // I2C 設備標頭檔
#include <MMA8653.h> // 加速度感應器標頭檔
#include <MPU6050.h>        // 陀螺儀標頭檔
#include <IRremoteForStuduino.h>        // 紅外接收器標頭檔
#include <ColorSensor.h>    // 顏色感應器標頭檔
#include "Studuino.h"// Studuino 標頭檔

// Studuino 板圖像。 每個程序只製作一個。
Studuino board;

// 在程序開始時調用一次。 主要用於初始化。
void setup() {
   // 使用初始化功能初始化連接部件的 Studuino 端口
   board.InitDCMotorPort(PORT_D10);     // 初始化端口 D10 以用於伺服馬達
   board.InitDCMotorPort(PORT_D11);     // 初始化端口 D11 以用於伺服馬達
   board.InitDCMotorPort(PORT_D12);     // 初始化端口 D12 以用於伺服馬達
}
// 這個功能運行一個無限循環。 如：主要流程。
void loop() {
   // 初始化伺服馬達角度為 90 度
   byte connector[] = { PORT_D10, PORT_D11, PORT_D12 };
   byte degree[] = { 90, 90, 90 };
   byte number = sizeof(connector) / sizeof(byte); // 將設置角度的端口數

   board.AsyncServomotors(connector, degree, number);
   // 用家可以使用它來添加延遲，直到伺服馬達旋轉
   board.Timer(1000);

   board.Timer(3000);          // 等待 3 秒

   // 將連接到端口 D10、D11 和 D12 的伺服馬達設置為 90、180 和 0 度
   degree[0] = 90;
   degree[1] = 180;
   degree[2] = 0;

   // 一旦此功能完成運行，伺服馬達將達到其目標角度

SyncServomotors(connector, degree, number, 10);

   board.Timer(3000);          // 等待 3 秒

   // 將連接到端口 D10 的伺服馬達設置為 180 度
   board.Servomotor(PORT_D10, 180);
   // 用家可以使用它在他們的伺服電機旋轉之前添加延遲
   board.Timer(1000);

   for (;;) {}         // 無限循環以防止程序從頂部重新開始
}
```

3.4.3. 蜂鳴器

將蜂鳴器連接到 Studuino 上的 A0 並使用 Arduino IDE 加載以下程序。 該程序播放“Do”一秒鐘，播放“So”一秒鐘，然後播放 Twinkle、Twinkle、Little Star。

```
#include <Arduino.h >// 基本標頭檔
#include <Servo.h>              // 伺服馬達標頭檔
#include <Wire.h>               // I2C 設備標頭檔
#include <MMA8653.h> // 加速度感應器標頭檔
#include <MPU6050.h>        // 陀螺儀標頭檔
#include <IRremoteForStuduino.h>        // 紅外接收器標頭檔
#include <ColorSensor.h>    // 顏色感應器標頭檔
#include "Studuino.h"// Studuino 標頭檔

// Studuino 板圖像。 每個程序只製作一個。
Studuino board;

// 在程序開始時調用一次。 主要用於初始化。
void setup() {
   // 使用初始化功能初始化連接部件的 Studuino 端口
   board.InitDCMotorPort(PORT_A0 ,DC ); // 初始化端口 A0 以用於蜂鳴器
}

// 這個功能運行一個無限循環。 如：主要流程。
void loop() {
   // 從蜂鳴器播放一秒音符
   board.Buzzer(PORT_A0, BZR_C5, 1000);

   board.Timer(1000);          // 等待 1 秒

   // 從蜂鳴器播放一秒音符
   board.BuzzerControl(PORT_A0, ON, BZR_G5);
   board.Timer(1000);
   board.BuzzerControl(PORT_A0, OFF, 0);// 設置為 OFF 時忽略最後一個參數

   board.Timer(1000);          // 等待 1 秒

   // 從蜂鳴器中播放一段旋律
   word myPitches[] = { BZR_C5, BZR_C5, BZR_G5, BZR_G5, BZR_A5, BZR_A5, BZR_G5 };
   byte number = sizeof(myScales) / sizeof(word); // 要播放的音符數
   float myBeats[] = { 1, 1, 1, 1, 1, 1, 1 };     // 節奏
   board.Melody (PORT_A0, myPitches, myBeats, number, TEMPO90);

   for (;;) {}        // 無限循環以防止程序從頂部重新開始
}
```

3.4.4. LED

將 LED 連接到 Studuino 上的 A1 和 D9 並使用 Arduino IDE 加載以下程序。 LED A1 將閃爍 3 次，然後 LED D9 緩慢亮起。

```
#include <Arduino.h >// 基本標頭檔
#include <Servo.h>              // 伺服馬達標頭檔
#include <Wire.h>               // I2C 設備標頭檔
#include <MMA8653.h> // 加速度感應器標頭檔
#include <MPU6050.h>        // 陀螺儀標頭檔
#include <IRremoteForStuduino.h>        // 紅外接收器標頭檔
#include <ColorSensor.h>    // 顏色感應器標頭檔
#include "Studuino.h" // Studuino 標頭檔

// Studuino 板圖像。 每個程序只製作一個。
Studuino board;

// 在程序開始時調用一次。 主要用於初始化。
void setup() {
  // 使用初始化功能初始化連接部件的 Studuino 端口
  board.InitDCMotorPort(PORT_A1 ,DC ); // 初始化端口 A0 以用於 LED
  board.InitDCMotorPort(PORT_D9);       // 初始化端口 D9 以用於 LED
}

// 這個功能運行一個無限循環。 如：主要流程。
void loop() {
  // 閃爍 LED A1 3 次
  for (int i = 0;i < 3;i++) {
    board.LED(PORT_A1, ON);   // 打開 LED A1
    board.Timer(1000);        // 等待 1 秒
    board.LED(PORT_A1, OFF);  // 打開 LED A1
    board.Timer(1000);        // 等待 1 秒
  }

  // 緩慢打開 LED D9
  board.Gradation(PORT_D9, 0);
  for (int i = 0;i < 255;i++) {
    board.Gradation(PORT_D9, i);
    board.Timer(100);
  }

  for (;;) {}         // 無限循環以防止程序從頂部重新開始
}
```

3.4.5. 感應器

① 常規感應器

將觸摸感應器連接到 A1，將聲音感應器連接到 A2，將紅外光反射器連接到 A3，將加速度感應器連接到 A4/A5，將光感應器連接到 A6，然後加載以下程序。 加載程序後，轉到 Arduino IDE 中的工具並選擇串行監視器以打開串行監視器，它會顯示每個傳感器的值。

```
#include <Arduino.h >// 基本標頭檔
#include <Servo.h>              // 伺服馬達標頭檔
#include <Wire.h>               // I2C 設備標頭檔
#include <MMA8653.h> // 加速度感應器標頭檔
#include <MPU6050.h>        // 陀螺儀標頭檔
#include <IRremoteForStuduino.h>        // 紅外接收器標頭檔
#include <ColorSensor.h>    // 顏色感應器標頭檔
#include "Studuino.h" // Studuino 標頭檔

// Studuino 板圖像。 每個程序只製作一個。
Studuino board;

// 在程序開始時調用一次。 主要用於初始化。
void setup() {
   // 使用初始化功能初始化連接部件的 Studuino 端口
   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);

   // 開始串行輸出
   Serial.begin(9600);
}

// 這個功能運行一個無限循環。 如：主要流程。
void loop() {
   // 每 100 毫秒檢索一次光感應器值並輸出到串行監視器
   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);
   }
}
```

② 可選配件：超聲波感應器、溫度感應器、陀螺儀

將超聲波感應器連接到 A0/A1，將溫度感應器連接到 A2，將陀螺儀連接到 A4/A5，然後使用 Arduino IDE 加載以下程序。 加載程序後，轉到 Arduino IDE 中的工具並選擇串行監視器以打開串行監視器。 串行監視器向您顯示每個傳感器的值。

```
#include <Arduino.h >// 基本標頭檔
#include <Servo.h>              // 伺服馬達標頭檔
#include <Wire.h>               // I2C 設備標頭檔
#include <MMA8653.h> // 加速度感應器標頭檔
#include <MPU6050.h>        // 陀螺儀標頭檔
#include <IRremoteForStuduino.h>        // 紅外接收器標頭檔
#include <ColorSensor.h>    // 顏色感應器標頭檔
#include "Studuino.h"// Studuino 標頭檔

// Studuino 板圖像。 每個程序只製作一個。
Studuino board;

// 在程序開始時調用一次。 主要用於初始化。
void setup() {
   // 使用初始化功能初始化連接部件的 Studuino 端口
   board.InitSensorPort(PORT_A0, PORT_A1, PIDULTRASONICSENSOR);
   board.InitSensorPort(PORT_A2, PIDTEMPERATURESENSOR);
   board.InitI2CPort(PIDGYROSCOPE);

   // 開始串行輸出
   Serial.begin(9600);
}

// 這個功能運行一個無限循環。 如：主要流程。
void loop() {
   // 每 100 毫秒檢索一次光感應器值並輸出到串行監視器
   int uVal = board.GetUltrasonicSensorValue(PORT_A0, PORT_A1);
   int tVal = board.GetTemperatureSensorValue(PORT_A2);
   int xVal = board.GetGyroscopeValue(X_AXIS);
   int yVal = board.GetGyroscopeValue(Y_AXIS);
   int zVal = board.GetGyroscopeValue(Z_AXIS);
   int gxVal = board.GetGyroscopeValue(GX_AXIS);
   int gyVal = board.GetGyroscopeValue(GY_AXIS);
   int gzVal = board.GetGyroscopeValue(GZ_AXIS);
   Serial.print("ultrasonic:");         Serial.print(uVal);         Serial.print("¥t");
   Serial.print("temperature:");        Serial.print(tVal);         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("gx:");        Serial.print(gxVal);        Serial.print("¥t");
   Serial.print("gy:");        Serial.print(gyVal);        Serial.print("¥t");
   Serial.print("gz:");        Serial.print(gzVal);        Serial.println();;
   board.Timer(100);
}
```

③ 可選部件：**IR** 接收器

將 IR 接收器連接到 Studuino 上的 A0 並使用 Arduino IDE 加載以下程序。 加載程序後，轉到 Arduino IDE 中的工具並選擇串行監視器以打開串行監視器。 串行監視器向您顯示每個感應器的值。

```
#include <Arduino.h >// 基本標頭檔
#include <Servo.h>              // 伺服馬達標頭檔
#include <Wire.h>               // I2C 設備標頭檔
#include <MMA8653.h> // 加速度感應器標頭檔
#include <MPU6050.h>        // 陀螺儀標頭檔
#include <IRremoteForStuduino.h>        // 紅外接收器標頭檔
#include <ColorSensor.h>    // 顏色感應器標頭檔
#include "Studuino.h"// Studuino 標頭檔

// Studuino 板圖像。 每個程序只製作一個。
Studuino board;

// 在程序開始時調用一次。 主要用於初始化。
void setup() {
   // 使用初始化功能初始化連接部件的 Studuino 端口
   board.InitSensorPort(PORT_A0, PIDIRRECEIVER);  // 感應器輸入設置

   // 開始串行輸出
   Serial.begin(9600);
}

// 這個功能運行一個無限循環。 如：主要流程。
void loop() {
   // 每 100 毫秒檢索一次光感應器值並輸出到串行監視器
   unsigned long ir = board.GetIRReceiverValue();
   if(ir != 0) {
     Serial.print("IR receive:");       Serial.print(ir, HEX);      Serial.println();
   }
   board.Timer(100);
}
```

④ 可選部件：顏色感應器

將顏色感應器連接到 Studuino 上的 A4/A5 並使用 Arduino IDE 加載以下程序。 加載程序後，轉到 Arduino IDE 中的工具並選擇串行監視器以打開串行監視器。 串行監視器向您顯示每個感應器的值。

```
#include <Arduino.h >// 基本標頭檔
#include <Servo.h>              // 伺服馬達標頭檔
#include <Wire.h>               // I2C 設備標頭檔
#include <MMA8653.h> // 加速度感應器標頭檔
#include <MPU6050.h>        // 陀螺儀標頭檔
#include <IRremoteForStuduino.h>        // 紅外接收器標頭檔
#include <ColorSensor.h>    // 顏色感應器標頭檔
#include "Studuino.h"// Studuino 標頭檔

// Studuino 板圖像。 每個程序只製作一個。
Studuino board;

// 在程序開始時調用一次。 主要用於初始化。
void setup() {
   // 使用初始化功能初始化連接部件的 Studuino 端口
   board.InitI2CPort(PIDCOLORSENSOR);  // 啟動顏色感應器

   // 開始串行輸出
   Serial.begin(9600);
}

// 這個功能運行一個無限循環。 如：主要流程。
void loop() {
   // 每 100 毫秒檢索一次光感應器值並輸出到串行監視器
   unsigned int rVal = board.GetColorSensorValue(VALUE_RED);
   unsigned int gVal = board.GetColorSensorValue(VALUE_GREEN);
   unsigned int bVal = board.GetColorSensorValue(VALUE_BLUE);
   unsigned int cVal = board.GetColorSensorValue(VALUE_CLEAR);
   double x, y;
   board.GetColorSensorXY(&x, &y);

   Serial.print("red:");       Serial.print(rVal);         Serial.print("¥t");
   Serial.print("green:");     Serial.print(gVal);         Serial.print("¥t");
   Serial.print("blue:");      Serial.print(bVal);         Serial.print("¥t");
   Serial.print("clear:");     Serial.print(cVal);         Serial.print("¥t");
   Serial.print("X:");         Serial.print(x);   Serial.print("¥t");
   Serial.print("Y:");         Serial.print(y);   Serial.println();
   board.Timer(100);
}
```

⑤ 可選配件：藍牙模組

在不連接任何東西的情況下，使用 Arduino IDE 加載以下程序。 加載後，將 D0 和 D1 上的藍牙通信引腳插入 A0-A7 中任何打開的傳感器端口的電源。 打開應用程序，連接您的 Studuino，並發送值以打開和關閉引腳 13 上 Studuino 的板載 LED。

```
#include <Arduino.h >// 基本標頭檔
#include <Servo.h>              // 伺服馬達標頭檔
#include <Wire.h>               // I2C 設備標頭檔
#include <MMA8653.h> // 加速度感應器標頭檔
#include <MPU6050.h>        // 陀螺儀標頭檔
#include <IRremoteForStuduino.h>        // 紅外接收器標頭檔
#include <ColorSensor.h>    // 顏色感應器標頭檔
#include "Studuino.h"// Studuino 標頭檔

// Studuino 板圖像。 每個程序只製作一個。
Studuino board;

// 在程序開始時調用一次。 主要用於初始化。
void setup() {
   // 使用初始化功能初始化連接部件的 Studuino 端口
   board.InitBluetooth();
   pinMode(13, OUTPUT);
}

// 這個功能運行一個無限循環。 如：主要流程。
void loop() {
   // 從應用程序中獲取值
   board.UpdateBluetooth();

   // 根據接收到的值打開或關閉 LED
   if(board.GetBTCommandIDState(BT_ID_01)) {
     digitalWrite(13, HIGH);
   }
   if(board.GetBTCommandIDState(BT_ID_02)) {
     digitalWrite(13, LOW);
   }
   board.Timer(100);
}
```

A. 將直流馬達連接到 Studuino

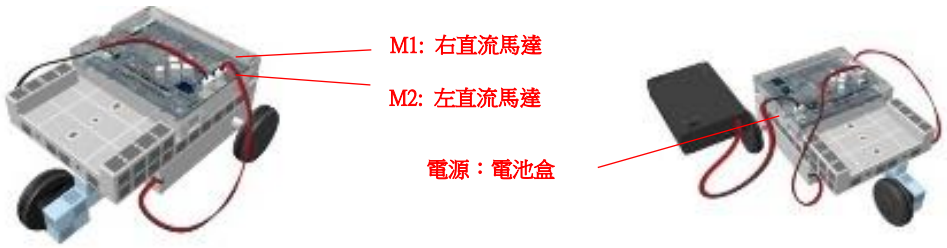
請按照以下說明組裝您的汽車：

(1) 如下圖所示將輪子安裝到直流馬達上。
★ 做對稱的一對。

(2) 將兩個直流馬達連接到 Studuino 支架的底部。

(3) 使用積木製作後輪。

(4) 現在將您的直流馬達和電池盒插入您的 Studuino。



(5) 將電池盒安裝在 Studuino 支架上以保持其固定。