Bibliothèque Studuino

Référence des fonctions

Ce manuel est un guide de référence pour les fonctions de la bibliothèque Studuino. L'environnement de programmation Studuino étant en développement, ce manuel peut être amené à être modifié ou révisé.

Sommaire

1. À propos de la bibliothèque de fonctions Studuino 2
2. Fonction 2
2.1. Fonction d'initialisation 2
2.2. Fonction de contrôle de moteur à courant continu 4
2.3. Fonction de contrôle de servomoteur 6
2.4. Fonction de contrôle d'un avertisseur sonore 8
2.5. Fonction de contrôle d'une DEL 9
2.6. Fonction d'entrée de données 10
2.7. Fonction minuteur 12
3. Programmation 15
3.1. Language Arduino 15
3.2. Objet Studuino 15
3.3. Inclure un fichier d'en-tête (Header File) 15
3.4. Exemples de programmes 16
3.4.1. Exemple de programme pour moteur à courant continu 16
3.4.2. Exemple de programme pour servomoteurs 17
3.4.3. Exemple de programme pour un avertisseur sonore 18
3.4.4. Exemple de programme pour DEL 19
3.4.5. Exemples de programme utilisant des capteurs 20
Annexe A. Raccordement de la carte Studuino et d'un moteur à courant continu 23

1. À propos de la bibliothèque de fonctions Studuino

La bibliothèque de fonctions Studuino est utilisée pour communiquer avec votre Studuino à l'aide de l'Arduino IDE. La bibliothèque vous permet de contrôler facilement les pièces, car les niveaux de chaque pièce (moteur à courant continu, servomoteur, etc.) sont déjà configurés pour vous.

2. Fonction

L'explication de chaque fonction sera présentée selon le format suivant :

Nom de la fonction	**(Nom de la fonction)**			
Argument	(Type)	(Nom de la variable)	(Valeur)	(Explication)
Retours	(Type)	(Explication)		
Notes				

La section suivante décrit chaque fonction.

2.1. Fonction d'initialisation

La section suivante décrit les fonctions qui sont utilisées pour initialiser les ports sur votre Studuino.

Nom de la fonction	**InitDCMotorPort**			
Argument	Octet	Connecteur	PORT_M1 PORT_M2	Connecteur
Retours	Sans objet			

Utilisez cette fonction pour initialiser un port de moteur à courant continu avant d'utiliser un moteur à courant continu.
(Exemple)
//Utilisez cette fonction pour initialiser un port de moteur à courant continu avant d'utiliser les fonctions Move, DCMotor, DCMotorPower ou DCMotorControl
InitDCMotorPort(PORT_M1); //Initialise un moteur à courant continu raccordé au port M1				

Nom de la fonction	**InitServomotorPort**			
Argument	Octet	Connecteur	Voir nº 1	Connecteur
Retours	Sans objet			

Utilisez cette fonction pour initialiser un port de servomoteur avant d'utiliser un servomoteur.
(Exemple)
// Utilisez cette fonction pour initialiser le port de servomoteur avant d'utiliser les fonctions Servomotor, SyncServomotors ou AsyncServotomors.
InitServomotorPort(PORT_D2); // Initialise un servomoteur connecté au port D2				

Nom de la fonction	**InitServomotorPortForLED**			
Argument	Octet	Connecteur	PORT_D9	Connecteur
			PORT_D10	
			PORT_D11	
Retours	Sans objet			

Cette fonction initialise les ports servomoteurs D9, D10 et D11 pour une utilisation avec des DEL. Utilisez cette fonction pour initialiser un port avant d'essayer de contrôler la luminosité d'une DEL sur le port, D9, D10 et D11.
(Exemple)
// Utilisez cette fonction pour initialiser le port avant d'utiliser la fonction Gradation.
InitServomotorPortForLED (PORT_D9); // Initialise une DEL connectée au port D9
Gradation (PORT_D9, 128);				

Nom de la fonction	**InitSensorPort**			
Argument	Octet	Connecteur	Voir nº 4	Connecteur
	Octet	pid	Voir nº 5	Pièces reliées
Retours	Sans objet			
Utilisez cette fonction pour initialiser un port capteur/DEL/avertisseur sonore avant d'essayer d'utiliser un capteur, une DEL ou un avertisseur sonore. (Exemple) // Utilisez cette fonction pour initialiser un port avant d'utiliser les fonctions Buzzer, BuzzerControl, Melody, LED (DEL) ou Get *				

InitSensorPort (PORT_A0, PIDLED); // Initialise une DEL connectée au port A0

2.2. Fonction de contrôle de moteur à courant continu

La section suivante décrit les fonctions de contrôle des moteurs à courant continu.

Nom de la fonction	**Move (déplacement)**			
Argument	Octet	Direct	FORWARD	Vers l'avant
			BACKWARD	Reculer
			FORWARD_RIGHT	Virage à droite (avant)
			FORWARD_LEFT	Virage à gauche (avant)
			BACKWARD_RIGHT	Virage à droite (arrière)
			BACKWARD_LEFT	Virage à gauche (arrière)
			CLOCKWISE	Rotation vers la droite (dans le sens horaire)
			COUNTERCLOCKWISE	Rotation vers la gauche (dans le sens anti-horaire)
	Octet	Rythme	0～255	Vitesse (niveau)
	ULong	Durée	0～2^32-1	Temps (msec)
	Octet	Frein	BRAKE	Enclencher le frein
			COAST	Relâcher le frein
Retours	Sans objet			
Ces fonctions sont utilisées pour contrôler le mouvement d'une voiture à deux moteurs à courant continu. La performance de ces fonctions dépend de la façon dont les moteurs à courant continu sont connectés à votre Studuino. Construire une voiture en utilisant le guide en annexe **A. Raccordement de moteurs CC à Studuino.** (Exemple) Move (FORWARD, 10, 1000, BRAKE); // la voiture va continuer avec une vitesse de niveau 10 pendant une seconde, puis s'arrêter.				

Nom de la fonction	**DCMotor**			
Argument	Octet	Connecteur	PORT_M1	Connecteur
			PORT_M2	
	Octet	Rotation	NORMAL	Rotation vers la droite
			REVERSE	Rotation vers la gauche
	Octet	Rythme	0～255	Vitesse (niveau)
	ULong	Durée	0～2^32-1	Temps (msec)
	Octet	Frein	BRAKE	Enclencher le frein
			COAST	Relâcher le frein
Retours	Sans objet			

Cette fonction commande un moteur à courant continu.

(Exemple)

// Le moteur à courant continu connecté au port M1 tournera à une vitesse de niveau 10 pendant une seconde, puis s'arrêtera.

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

Nom de la fonction	**DCMotorPower**			
Argument	Octet	Connecteur	PORT_M1	Connecteur
			PORT_M2	
	Octet	Rythme	0～255	Vitesse (niveau)
Retours	Sans objet			

Cette fonction contrôle la vitesse d'un moteur à courant continu.

(Exemple)

// Le moteur à courant continu connecté au port M1 tournera à une vitesse de niveau 10 pendant une seconde, puis s'arrêtera. Ensuite, il tournera à 100 pendant une seconde, puis s'arrêtera.

DCMotorPower (PORT_M1, 10); // fixe la vitesse du moteur à courant continu raccordé en M1. DCMotorControl (PORT_M1, CLOCKWISE); // fait avancer le moteur à courant continu connecté en M1.

Timer(1000); compte une seconde.

DCMotorPower (PORT_M1, 100); change la vitesse du moteur à courant continu en M1

Timer(1000); compte une seconde.

DCMotorControl (PORT_M1, BRAKE); // arrête le moteur à courant continu connecté en M1

Nom de la fonction	**DCMotorControl**			
Argument	Octet	Connecteur	PORT_M1	Connecteur
			PORT_M2	
	Octet	Rotation	NORMAL	Rotation vers la droite
			REVERSE	Rotation vers la gauche
			BRAKE	Enclencher le frein
			COAST	Relâcher le frein
Retours	Sans objet			

Cette fonction contrôle la rotation d'un moteur à courant continu.

(Exemple)

// Le moteur à courant continu connecté en M1 tournera à une vitesse de niveau 10 pendant une seconde, puis s'arrêtera.

DCMotorPower (PORT_M1, 10); // fixe la vitesse du moteur à courant continu raccordé en M1.

DCMotorControl(PORT_M1, CLOCKWISE); // fait avancer le moteur à courant continu connecté en M1.

Timer(1000); compte une seconde.

DCMotorControl (PORT_M1, BRAKE); // arrête le moteur à courant continu connecté en M1

2.3. Fonction de contrôle de servomoteur

La section suivante décrit les fonctions de contrôle des servomoteurs.

Nom de la fonction	**Servomotor**			
Argument	Octet	Connecteur	Voir nº 1	Connecteur
	Octet	Degré	0～180	Angle du servomoteur
Retours	Sans objet			

Cette fonction définit l'angle d'un servomoteur. Cette fonction vous permet d'utiliser un autre processus, pendant que l'angle est défini.

(Exemple)

// Définit l'angle du servomoteur connecté en D2 à 90°.

Servomoteur (PORT_D2, 90);

Nom de la fonction	**AsyncServomotors**			
Argument	Octet	Connecteurs	Voir nº 1	Ordre des connecteurs
	Octet	Degrés	0～180	Degré de chaque servomoteur
	Octet	Nombre	1～8	Nombre de servomoteurs
Retours	Sans objet			

Cette fonction définit l'angle de plusieurs servomoteurs. Cette fonction vous permet d'utiliser un autre processus, pendant que les angles sont définis.

(Exemple)

```
// Définit chaque servomoteur connecté en D2, D9 et D10 à 90°, 180° et 45°.
byte myConectors[] = { PORT_D2, PORT_D9, PORT_D10 };
byte myDegrees[] = { 90, 180, 45};
ASyncServomotor (myConectors, myDegrees, 3);
```

Nom de la fonction	**SyncServomotors**			
Argument	Octet	Connecteur	Voir nº 1	Ordre des connecteurs
	Octet	Degré	0～180	Degré de chaque servomoteur
	Octet	Nombre	1～8	Nombre de servomoteurs
	Octet	Temps	0～255	Temps de mouvement (en msec) par 1°
Retours	Sans objet			

Cette fonction définit l'angle de plusieurs servomoteurs. Cette fonction n'effectuera aucune action jusqu'à ce que l'angle pour le servomoteur ait été défini. Utilisez l'argument du temps pour changer la vitesse rapide ou lente avec laquelle l'angle est défini. Toutefois, si la durée programmée est inférieure à 3, la vitesse de mouvement du servomoteur ne changera pas, 3 msec étant la vitesse de pointe pour un changement de 1°.

(Exemple)

```
// Définit chaque servomoteur connecté en D2, D9 et D10 à 90°, 180° et 45°.
byte myConectors[] = { PORT_D2, PORT_D9, PORT_D10 };
byte myDegrees[] = { 90, 180, 45};
ASyncServomotor (myConectors, myDegrees, 3, 5);
```

2.4. Fonction de contrôle d'un avertisseur sonore

La section suivante décrit les fonctions de contrôle des avertisseurs sonores (buzzers).

Nom de la fonction	**Buzzer**			
Argument	Octet	Connecteur	Voir n°3	Connecteur
	Mot	Ton	Voir n°6	Ton du son
	ULong	Durée	0～2^32-1	Temps d'entrée (msec)
Retours	Sans objet			
Joue une note désignée pour la durée spécifiée. (Exemple) Buzzer (PORT_A0, BZR_C4, 1000); fait jouer la note « Do » à l'avertisseur sonore connecté en A0 pendant une seconde.				

Nom de la fonction	**BuzzerControl**			
Argument	Octet	Connecteur	Voir n°3	Connecteur
	Booléen	Allumé/éteint	ON	Sortie audio
			Désactivé	Arrête le son
	Octet	Ton	Voir n°6	Ton
Retours	Sans objet			
Lorsque l'avertisseur sonore est réglé sur ON (allumé), l'avertisseur sonore jouera la note désignée. Lorsque l'avertisseur sonore est réglé sur OFF (éteint), il n'y a aucun son (la valeur du ton désigné sera de zéro). (Exemple) // fait jouer la note « Do » à l'avertisseur sonore connecté en A0 pendant une seconde. BuzzerControl (PORT_A0, ON, BZR_C4); Timer(1000); BuzzerControl (PORT_A0, OFF, BZR_C4);				

Nom de la fonction	**Melody**

Argument	Octet	Connecteur	Voir nº3	Connecteur
	Mot	Tons	Voir nº6	Ton
	float[]	Rythmes	0～	Rythme
	Octet	Nombre	Vitesse de mélodie (0 - 225)	Nombre de notes
	Octet	Tempo	TEMPO60	Tempo
			TEMPO90	
			TEMPO120	
			TEMPO150	
Retours	Sans objet			

L'avertisseur sonore joue une mélodie programmée.

(Exemple)

```
// Fait jouer la mélodie « Do Ré Mi Fa Mi Ré Do » à l'avertisseur sonore connecté en 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; // nombre d'éléments
Melody (PORT_A0, myPitches, myBeats, num, TEMPO90);
```

Des mélodies plus longues peuvent dépasser la quantité de mémoire SRAM sur la carte Studuino. Lorsque l'espace de données a dépassé l'espace sur votre SRAM, tapez le mot clé PROGMEM et vous pouvez enregistrer et utiliser l'espace dans l'espace de données de la mémoire Flash. Vous pouvez afficher la page Internet ci-dessous pour plus d'informations en tapant le mot clé **PROGMEM**.

https://www.arduino.cc/en/Reference/PROGMEM

2.5. Fonction de contrôle d'une DEL

La section suivante décrit les fonctions de contrôle des DEL.

Nom de la fonction	**DEL**			
Argument	Octet	Connecteur	Voir nº3	Connecteur
	Booléen	Allumé/éteint	ON	DEL ON/OFF (allumée/éteinte)
			OFF	
Retours	Sans objet			

Allume et éteint la DEL. (Exemple) LED (PORT_A0, ON); // allume la DEL connectée en A0.

Nom de la fonction	**Gradation**			
Argument	octet	Connecteur	PORT_D9 PORT_D10 PORT_D11	Connecteur
	octet	ratio	0～255	Luminosité (devient plus lumineuse lorsque la valeur de la luminosité est plus grande)
Retours	sans objet			
Utilisez cette fonction pour régler la luminosité des DEL connectées en D9, D10 et D11. (Exemple) Gradation (PORT_D9, 128); règle la luminosité de la DEL connectée en D9.				

2.6. Fonction d'entrée de données

La section suivante décrit les fonctions des boutons poussoirs et des capteurs Studuino.

Nom de la fonction	**GetPushSwitchValue**			
Argument	Octet	Connecteur	Voir n°2	Connecteur
Retours	Octet (byte)	0 : enfoncé, 1 : relâché		

Vérifie l'état de l'interrupteur à poussoir
(Exemple)
// Vérifiez la valeur de l'interrupteur à poussoir connecté en A0
byte val = GetPushSwitchValue (PORT_A0);				

Nom de la fonction	**GetTouchSensorValue**			
Argument	Octet	Connecteur	Voir n°3	Connecteur
Retours	Octet	0 : enfoncé, 1 : relâché		

	(byte)	

Vérifie la valeur du capteur de pression
(Exemple)
// Vérifiez la valeur du capteur de pression connecté en A0
byte val = GetTouchSensorValue (PORT_A0);		

Nom de la fonction	**GetLightSensorValue**			
Argument	Octet	Connecteur	Voir nº 4	Connecteur
Retours	int	0～1023		
Vérifie la valeur du capteur de lumière (Exemple) int val = GetLightSensorValue (PORT_A0); // vérifie la valeur du capteur de lumière connecté en A0.				

Nom de la fonction	**GetSoundSensorValue**			
Argument	octet	Connecteur	Voir nº 4	Connecteur
Retours	int	0～1023		
Vérifie la valeur du capteur sonore (Exemple) int val = GetSoundSensorValue (PORT_A0); // vérifie la valeur du capteur sonore connecté en A0.				

Nom de la fonction	**GetIRPhotoreflectorValue**			
Argument	Octet	Connecteur	Voir nº 4	Connecteur
Retours	int	0～1023		

Vérifie la valeur du capteur infrarouge réflectif
(Exemple)
// Vérifie la valeur du capteur infrarouge réflectif connecté en A0
int val = GetIRPhotoreflectorValue (PORT_A0);				

Nom de la fonction	GetAccelerometerValue			
Argument	Octet	Axe	X_AXIS	L'axe que vous souhaitez mesurer
			Y_AXIS	
			Z_AXIS	
Retours	int	0～1023		
Vérifie la valeur de l'accéléromètre. Étant donné que l'accéléromètre utilise I2C, il est uniquement compatible avec les ports A4 et A5. (Exemple) int val = GetAccelerometerValue (X_AXIS); // vérifie l'inclinaison de l'axe X de l'accéléromètre.				

2.7. Fonction minuteur

La section suivante décrit la fonction minuteur (Timer).

Nom de la fonction	Timer			
Argument	ULong	Temps	0～2^32-1	Temps (msec)
Retours	Sans objet			
Compte pour la durée spécifiée (Exemple) Timer(1000); // compte une seconde				

*** 1 Réglages du port de servomoteur**

Valeur	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 Réglages du bouton poussoir**

Valeur	Port
PORT_A0	A0
PORT_A1	A1
PORT_A2	A2
PORT_A3	A3

***3 Réglages des ports analogiques**

Valeur	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

*** 4 Réglages des ports digitaux**

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

*** 5 Identifiant (ID) de la pièce**

Valeur	Partie
PIDOPEN	Déconnecté
PIDLED	DEL
PIDBUZZER	Avertisseur sonore
PIDLIGHTSENSOR	Capteur de lumière
PIDSOUNDSENSOR	Capteur sonore
PIDIRPHOTOREFLECTOR	Capteur infrarouge réflectif
PIDACCELEROMETER	Accéléromètre
PIDTOUCHSENSOR	Capteur de pression
PIDPUSHSWITCH	Interrupteur à poussoir

*** 6 Valeur du ton**

Valeur	Échelle	Hz
BZR_C3	Do	130
BZR_CS3	Do #	139
BZR_D3	Ré	147
BZR_DS3	Ré #	156
BZR_E3	Mi	165
BZR_F3	Fa	175
BZR_FS3	Fa #	185
BZR_G3	Sol	196
BZR_GS3	Sol #	208
BZR_A3	La	220

Valeur	Échelle	Hz
BZR_C5	Do	523
BZR_CS5	Do #	554
BZR_D5	Ré	587
BZR_DS5	Ré #	622
BZR_E5	Mi	659
BZR_F5	Fa	698
BZR_FS5	Fa #	740
BZR_G5	Sol	784
BZR_GS5	Sol #	831
BZR_A5	La	880

Valeur	Échelle	Hz
BZR_C7	Do	2093
BZR_CS7	Do #	2217
BZR_D7	Ré	2349
BZR_DS7	Ré #	2489
BZR_E7	Mi	2637
BZR_F7	Fa	2794
BZR_FS7	Fa #	2960
BZR_G7	Sol	3136
BZR_GS7	Sol #	3322
BZR_A7	La	3520

BZR_AS3	La #	233
BZR_B3	Si	247
BZR_C4	Do	262
BZR_CS4	Do #	277
BZR_D4	Ré	294
BZR_DS4	Ré #	311
BZR_E4	Mi	330
BZR_F4	Fa	349
BZR_FS4	Fa #	370
BZR_G4	Sol	392
BZR_GS4	Sol #	415
BZR_A4	La	440
BZR_AS4	La #	466
BZR_B4	Si	494

BZR_AS5	La #	932
BZR_B5	Si	988
BZR_C6	Do	1047
BZR_CS6	Do #	1109
BZR_D6	Ré	1175
BZR_DS6	Ré #	1245
BZR_E6	Mi	1319
BZR_F6	Fa	1397
BZR_FS6	Fa #	1480
BZR_G6	Sol	1568
BZR_GS6	Sol #	1661
BZR_A6	La	1760
BZR_AS6	La #	1865
BZR_B6	Si	1976

BZR_AS7	La #	3729
BZR_B7	Si	3951
BZR_C8	Do	4186
BZR_S	Pas de son	0

3. Programmation

La section suivante décrit les éléments que vous devez garder à l'esprit lors de la programmation à l'aide des fonctions de la bibliothèque Studuino.

3.1. Language Arduino

Vous devez être capable de définir la fonction de configuration et les fonctions loop (boucle) suivantes en langage Arduino. La fonction de configuration ne sera exécutée qu'une seule fois. Les processus définis dans la fonction loop seront répétés indéfiniment.

```
// Utilisée pour initialiser un programme. S'exécute seulement une fois.
void setup() {
// Utilisez les fonctions d'initialisation pour ouvrir sur Studuino des ports qui ont des pièces branchées.
}

// Cette fonction est répétée en continu et est le processus principal de votre programme.
void loop() {
}
```

3.2. Objet Studuino

Lorsque vous utilisez la bibliothèque Studuino, il est important que l'objet Studuino reflète l'image d'une carte Studuino en utilisant une variable globale.

```
// Reconnaît la carte Studuino. Ne l'incluez qu'une seule fois par programme
Studuino board;
```

3.3. Inclure un fichier d'en-tête (Header File)

Incluez des fichiers d'en-tête lors de l'utilisation des fonctions de la bibliothèque Studuino pour les servomoteurs et les accéléromètres. Ces fichiers d'en-tête doivent être inclus avec les fichiers d'en-tête Studuino.

```
#include <Arduino.h >          // Fichier d'en-tête de base
#include <Servo.h>             // Fichier d'en-tête pour servomoteur
#include <Wire.h>              // Fichier d'en-tête pour accéléromètre
#include <MMA8653.h>           // Fichier d'en-tête pour accéléromètre
#include "Stduino.h"           // Fichier d'en-tête pour Studuino
```

3.4. Exemples de programmes

La section suivante contient des exemples de programmes pour chaque pièce.

3.4.1. Exemple de programme pour moteur à courant continu

Connectez le moteur à courant continu en M1 et M2 de votre carte Studuino en utilisant comme référence **l'annexe A. Connexion de moteurs à courant continu à Studuino**. Envoyez le programme suivant à votre Studuino en utilisant l'Arduino IDE. Le moteur à courant continu connecté en M1 se déplacera vers l'avant pendant une seconde. La voiture se déplace vers l'avant pendant une seconde puis en marche arrière pendant une seconde.

```
#include <Arduino.h >          // Fichier d'en-tête de base
#include <Servo.h>             // Fichier d'en-tête pour servomoteur
#include <Wire.h>              // Fichier d'en-tête pour accéléromètre
#include <MMA8653.h>           // Fichier d'en-tête pour accéléromètre
#include   "Studuino.h"        // Fichier d'en-tête pour Studuino

// Reconnaît la carte Studuino. Ne l'incluez qu'une seule fois par programme.
Studuino board;

// Utilisée pour initialiser un programme. S'exécute seulement une fois.
void setup() {
// Utilisez les fonctions d'initialisation pour ouvrir des ports du Studuino qui ont des pièces branchées dessus
board.InitDCMotorPort(PORT_M1); // initialise le moteur à courant continu connecté en M1.
board.InitDCMotorPort(PORT_M2); // initialise le moteur à courant continu connecté en M2.
}

// Cette fonction est répétée en continu et est le processus principal de votre programme.
void loop() {
board.Move(FORWARD, 254, 1000, BRAKE);          // Se déplace vers l'avant pendant 1 seconde et s'arrête
board.Move(BACKWARD, 254, 1000, BRAKE);         // Se déplace en marche arrière pendant 1 seconde et s'arrête

// Le moteur à courant continu connecté en M1 se déplacera vers l'avant pendant une seconde, puis s'arrêtera.
board.DCMotorPower(PORT_M1, 254);               // Règle la vitesse de rotation du moteur à courant continu connecté en M1.
board.DCMotorControl(PORT_M1, NORMAL);          // Démarre le moteur DC connecté en M1
```

```
                                                    pour aller vers l'avant
board.Timer(1000);                                  // Attend une seconde
board.DCMotorControl(PORT_M1, BRAKE);  // Arrête le moteur à courant continu en M1

for (;;) {} // Insérez une boucle pour que le processus ne redémarre pas en partant du haut.
}
```

3.4.2. Exemple de programme pour servomoteurs

Connectez des servomoteurs en D10, D11 et D12. Envoyez le programme suivant à votre Studuino en utilisant l'Arduino IDE. Initialise tous les servomoteurs à 90° après trois secondes environ, les servomoteurs en D10, D11 et D12 seront réglés en même temps à 90°, 180° jusqu'à 0° à basse vitesse. Après trois secondes environ, le servomoteur en D10 sera réglé à 180°.

```
#include <Arduino.h >// Fichier d'en-tête de base
#include <Servo.h>              // Fichier d'en-tête pour servomoteur
#include <Wire.h>               // Fichier d'en-tête pour accéléromètre
#include <MMA8653.h>            // Fichier d'en-tête pour accéléromètre
#include   "Studuino.h"         // Fichier d'en-tête pour Studuino

// Reconnaît la carte Studuino. Ne l'incluez qu'une seule fois par programme.
Studuino board;

// Utilisée pour initialiser un programme. S'exécute seulement une fois.
void setup() {
// Utilisez les fonctions d'initialisation pour ouvrir des ports du Studuino qui ont des pièces branchées dessus
board.InitServomotorPort(PORT_D10); // initialise le servomoteur connecté en D10
board.InitServomotorPort(PORT_D11); // initialise le servomoteur connecté en D11
board.InitServomotorPort(PORT_D12); // initialise le servomoteur connecté en D12
}
// Cette fonction est répétée en continu et est le processus principal de votre programme.
void loop() {
    // Initialise les servomoteurs à 90°
    byte connector[] = { PORT_D10, PORT_D11, PORT_D12 };
    byte degree[] = { 90, 90, 90 };
    byte number = sizeof(connector) / sizeof(byte); // Nombre de degrés réglés pour chaque port
```

```
board.AsyncServomotors(connector, degree, number);
// Si nécessaire, vous pouvez entrer un retard dans le mouvement du servomoteur jusqu'à ce qu'il atteigne son angle de réglage.
board.Timer(1000);

board.Timer(3000);                    // Attendez trois secondes.

    // Règle les angles des servomoteurs en D10, D11 et D12 à 90°, 180° et 0°.
    degree[0] = 90;
    degree[1] = 180;
    degree[2] = 0;

// Le servomoteur se déplacera jusqu'à ce qu'il atteigne l'angle réglé après que cette fonction a été saisie
board. SyncServomotors(connector, degree, number, 10);

board.Timer(3000);                    // Attend trois secondes.

// Définit l'angle en degrés du servomoteur connecté en D10 à 180°
board.Servomotor(PORT_D10, 180);
// Si nécessaire, vous pouvez entrer un retard dans le mouvement du servomoteur jusqu'à ce qu'il atteigne son angle de réglage.
board.Timer(1000);


for (;;) {}        // Insérez une boucle pour que le processus ne redémarre pas en partant du haut.
}
```

3.4.3. Exemple de programme pour un avertisseur sonore

Connectez un avertisseur sonore en A0 et envoyez le programme suivant à votre Studuino en utilisant l'Arduino IDE. Après avoir joué « Do » et « Sol », l'avertisseur sonore jouera la mélodie de « ah vous dirai-je maman ».

```
#include <Arduino.h >// Fichier d'en-tête de base
#include <Servo.h>              // Fichier d'en-tête pour servomoteur
#include <Wire.h>               // Fichier d'en-tête pour accéléromètre
#include <MMA8653.h>            // Fichier d'en-tête pour accéléromètre
#include   "Studuino.h"         // Fichier d'en-tête pour Studuino

// Reconnaît la carte Studuino. Ne l'incluez qu'une seule fois par programme.
Studuino board;
```

```
// Utilisé pour initialiser un programme. S'exécute seulement une fois.
void setup() {
// Utilisez les fonctions d'initialisation pour ouvrir des ports du Studuino qui ont des pièces branchées
board.InitSensorPort(PORT_A0, PIDBUZZER);// initialise l'avertisseur sonore connecté en A0
}

// Cette fonction est répétée en continu et est le processus principal de votre programme.
void loop() {
// Arrête le son de l'avertisseur sonore pendant une seconde
board.Buzzer(PORT_A0, BZR_C5, 1000);

board.Timer(1000);                    // Attend une seconde

// Arrête le son de l'avertisseur sonore pendant une seconde
board.BuzzerControl(PORT_A0, ON, BZR_G5);
board.Timer(1000);
board.BuzzerControl(PORT_A0, OFF, 0);     // Le dernier argument sera ignoré si l'avertisseur sonore est éteint

board.Timer(1000);                    // Attend une seconde

// Joue la mélodie à partir de l'avertisseur sonore
word myPitches[] = { BZR_C5, BZR_C5, BZR_G5, BZR_G5, BZR_A5, BZR_A5, BZR_G5 };
byte number = sizeof(myScales) / sizeof(word); // Nombre de notes jouées
float [] myBeats = {1, 1, 1, 1, 1, 1, 1};          // Nombre de notes
board.Melody (PORT_A0, myPitches, myBeats, number, TEMPO90);

for (;;) {}       // Insérez une boucle pour que le processus ne redémarre pas en partant du haut.
}
```

3.4.4. Exemple de programme pour DEL

Connectez une DEL en A1 et A9 et envoyez le programme suivant à votre Studuino en utilisant l'Arduino IDE. La DEL en D9 clignotera lentement après que la DEL en A1 ait clignoté trois fois.

```
#include <Arduino.h >// Fichier d'en-tête de base
#include <Servo.h>              // Fichier d'en-tête pour servomoteur
#include <Wire.h>               // Fichier d'en-tête pour accéléromètre
#include <MMA8653.h>            // Fichier d'en-tête pour accéléromètre
#include   "Studuino.h"         // Fichier d'en-tête pour Studuino
```

```
// Reconnaît la carte Studuino. Ne l'incluez qu'une seule fois par programme.
Studuino board;

// Utilisée pour initialiser un programme. S'exécute seulement une fois.
void setup() {
// Utilisez les fonctions d'initialisation pour ouvrir des ports du Studuino qui ont des pièces branchées
board.InitSensorPort(PORT_A1, PIDLED);    // initialise la DEL connectée en A0
board.InitServomotorPortForLED(PORT_D9);// initialise la DEL connectée en D9
}

// Cette fonction est répétée en continu et est le processus principal de votre programme.
void loop() {
// La DEL connectée en A1 clignotera 3 fois
for (int i = 0;i < 3;i++) {
board.LED(PORT_A1, ON);          // La DEL en A1 clignote
board.Timer(1000);       // Attend une seconde
board.LED(PORT_A1, OFF);         // La DEL en A1 clignote
board.Timer(1000);       // Attend une seconde
    }

// La DEL connectée en D9 clignote lentement
board.Gradation(PORT_D9, 0);
for (int i = 0;i < 255;i++) {
board.Gradation(PORT_D9, i);
board.Timer(100);
}

for (;;) {}        // Insérez une boucle pour que le processus ne redémarre pas en partant du haut.
}
```

3.4.5. Exemples de programme utilisant des capteurs

Connectez le capteur de pression en A1, le capteur sonore en A2, le capteur infrarouge réflectif en A3, l'accéléromètre en A4/A5 et le capteur de lumière en A6. Envoyez le programme suivant à votre Studuino en utilisant l'Arduino IDE. Après avoir envoyé le programme, ouvrez Arduino IDE **Menu > Tools >** sélectionnez **Serial Monitor**. Le menu « Serial Monitor » affichera alors la valeur de chaque capteur.

```
#include <Arduino.h >// Fichier d'en-tête de base
#include <Servo.h>                // Fichier d'en-tête pour servomoteur
```

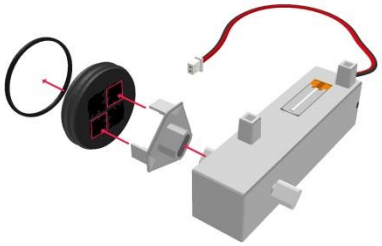
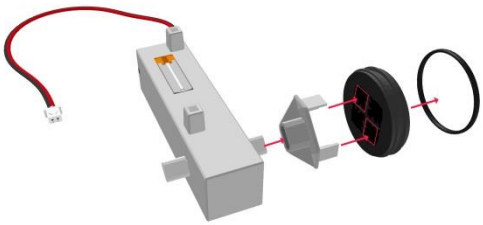
```
#include <Wire.h>                // Fichier d'en-tête pour accéléromètre
#include <MMA8653.h>             // Fichier d'en-tête pour accéléromètre
#include   "Studuino.h"          // Fichier d'en-tête pour Studuino

// Reconnaît la carte Studuino. Ne l'incluez qu'une seule fois par programme.
Studuino board;

// Utilisée pour initialiser un programme. S'exécute seulement une fois.
void setup() {
// Utilisez les fonctions d'initialisation pour ouvrir des ports du Studuino qui ont des pièces branchées
    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);

    // Initialise la sortie série
    Serial.begin(9600);
    }

// Cette fonction est répétée en continu et est le processus principal de votre programme.
void loop() {
    // Le moniteur série affichera la valeur du capteur de lumière en unités de 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);
    }
}
```

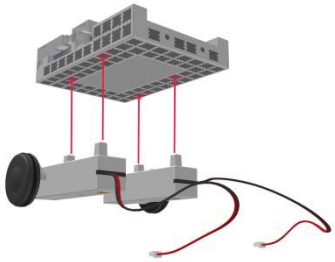
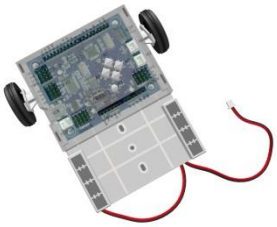
Annexe A.　Raccordement de la carte Studuino et d'un moteur à courant continu

Assemblez la voiture comme indiqué ci-dessous.

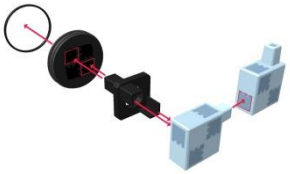
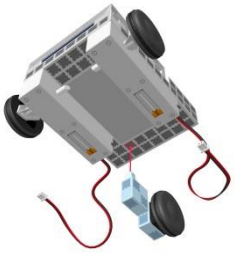
(1) Attachez les roues aux moteurs à courant continu.
★ Faites une paire symétrique

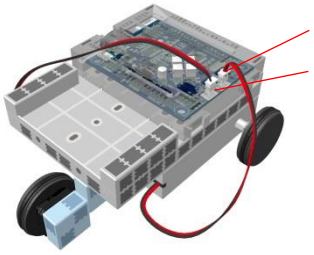
(2) Attachez les deux moteurs en dessous du support de la base.

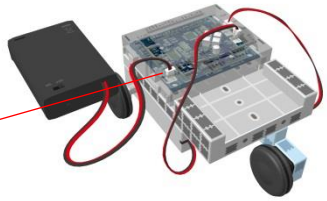



(3) Utilisez des blocs pour faire une roue arrière.


(4) Connectez les moteurs à courant continu et le boîtier de piles au circuit Studuino.





(5) Sécurisez le boîtier de piles au support de la base.