

Commencer

Merci d'avoir choisi les produits Freenove! Tout d'abord, lisez le document [About_Battery.pdf](#) dans le dossier décompressé.

Si vous n'avez pas téléchargé le fichier zip, veuillez le télécharger et le décompresser via le lien ci-dessous.

https://github.com/Freenove/Freenove_4WD_Car_Kit/archive/master.zip

Obtenez de l'aide et offrez des commentaires

Freenove fournit un support produit et technique gratuit et réactif, y compris mais sans s'y limiter:

- Problèmes de qualité des produits
- Utilisation du produit et problèmes de construction
- Questions concernant la technologie utilisée dans nos produits pour l'apprentissage et l'éducation Vos commentaires et opinions
- sont toujours les bienvenus
- Nous encourageons également vos idées et suggestions pour de nouveaux produits et des améliorations de produits. Pour tout ce qui précède, vous pouvez nous envoyer un e-mail à:

support@freenove.com

Sécurité et précautions

Veuillez suivre les précautions de sécurité suivantes lors de l'utilisation ou du stockage de ce produit:

- Gardez ce produit hors de la portée des enfants de moins de 6 ans.
- Ce produit ne doit être utilisé que sous la surveillance d'un adulte, car les jeunes enfants n'ont pas le jugement nécessaire concernant la sécurité et les conséquences d'une mauvaise utilisation du produit.
- Ce produit contient de petites pièces et pièces tranchantes. Ce produit contient des pièces conductrices d'électricité. Soyez prudent avec les pièces conductrices d'électricité à proximité ou autour des blocs d'alimentation, des batteries et des circuits sous tension.
- Lorsque le produit est allumé, activé ou testé, certaines pièces se déplaceront ou tourneront. Pour éviter les blessures aux mains et aux doigts, éloignez-les de toute pièce mobile!
- Il est possible qu'un circuit mal connecté ou en court-circuit puisse provoquer une surchauffe. Dans ce cas, débranchez immédiatement l'alimentation électrique ou retirez les piles et ne touchez à rien tant qu'il n'a pas refroidi! Lorsque tout est sûr et frais, consultez le didacticiel du produit pour identifier la cause.
- N'utilisez le produit que conformément aux instructions et aux directives de ce didacticiel, sinon des pièces pourraient être endommagées ou vous pourriez vous blesser.
- Stockez le produit dans un endroit frais et sec et évitez d'exposer le produit à la lumière directe du soleil. Après utilisation, coupez toujours l'alimentation et retirez ou débranchez les piles avant de les ranger.

Voiture et robot pour Raspberry Pi

Nous avons également des voitures et des kits de robots pour Raspberry Pi. Si cela vous intéresse, veuillez visiter notre site Web pour plus de détails.

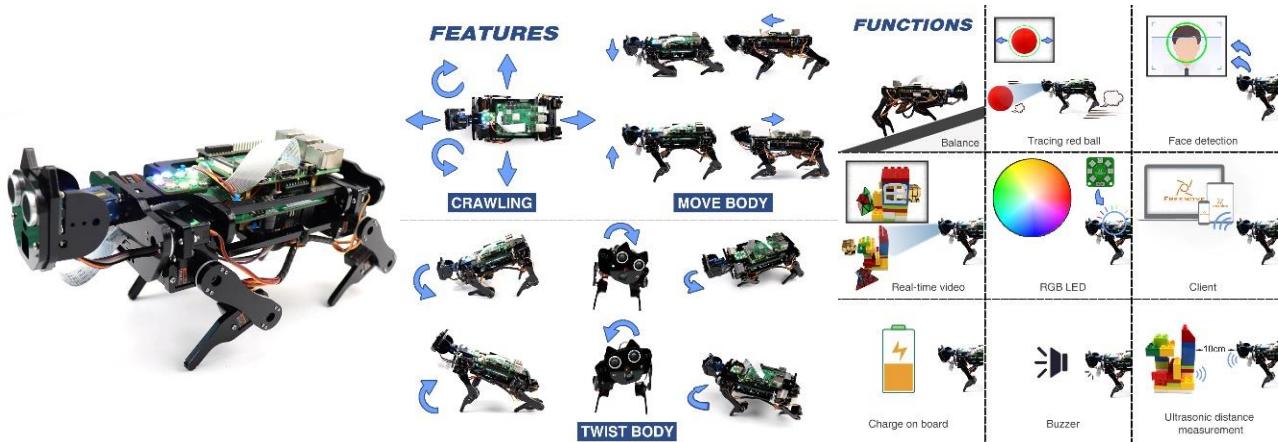
<http://www.freenove.com/store.html>

FNK0043 Kit voiture intelligente Freenove 4WD pour Raspberry Pi



<https://www.youtube.com/watch?v=4Zv0GZUQjZc>

FNK0050 Kit Chien Robot Freenove pour Raspberry Pi



https://www.youtube.com/watch?v=7BmIZ8_R9d4&t=35s

À propos de Freenove

Freenove fournit des produits et services électroniques open source dans le monde entier.

Freenove s'engage à accompagner ses clients dans leur formation à la robotique, à la programmation et aux circuits électroniques afin qu'ils puissent transformer leurs idées créatives en prototypes et en produits nouveaux et innovants. À cette fin, nos services incluent mais ne sont pas limités à:

- Kits de projets éducatifs et divertissants pour robots, voitures intelligentes et drones
- Kits éducatifs pour apprendre les systèmes logiciels robotiques pour Arduino, Raspberry Pi et micro:bit
- Assortiments de composants électroniques, modules électroniques et outils spécialisés
- Services de développement et de personnalisation de produits

Vous pouvez en savoir plus sur Freenove et obtenir nos dernières nouvelles et mises à jour via notre site Web:

<http://www.freenove.com>

droits d'auteur

Tous les fichiers, matériels et guides pédagogiques fournis sont publiés sous [Creative Commons AttributionNonCommercial-ShareAlike 3.0 Unported License](#). Une copie de cette licence se trouve dans le dossier contenant le didacticiel et les fichiers logiciels associés à ce produit.



Cela signifie que vous pouvez utiliser ces ressources dans vos propres œuvres dérivées, en partie ou complètement, mais **PAS dans l'intention ou à des fins d'utilisation commerciale.**

La marque et le logo Freenove sont la propriété de Freenove Creative Technology Co., Ltd. et ne peuvent pas être utilisés sans autorisation écrite.



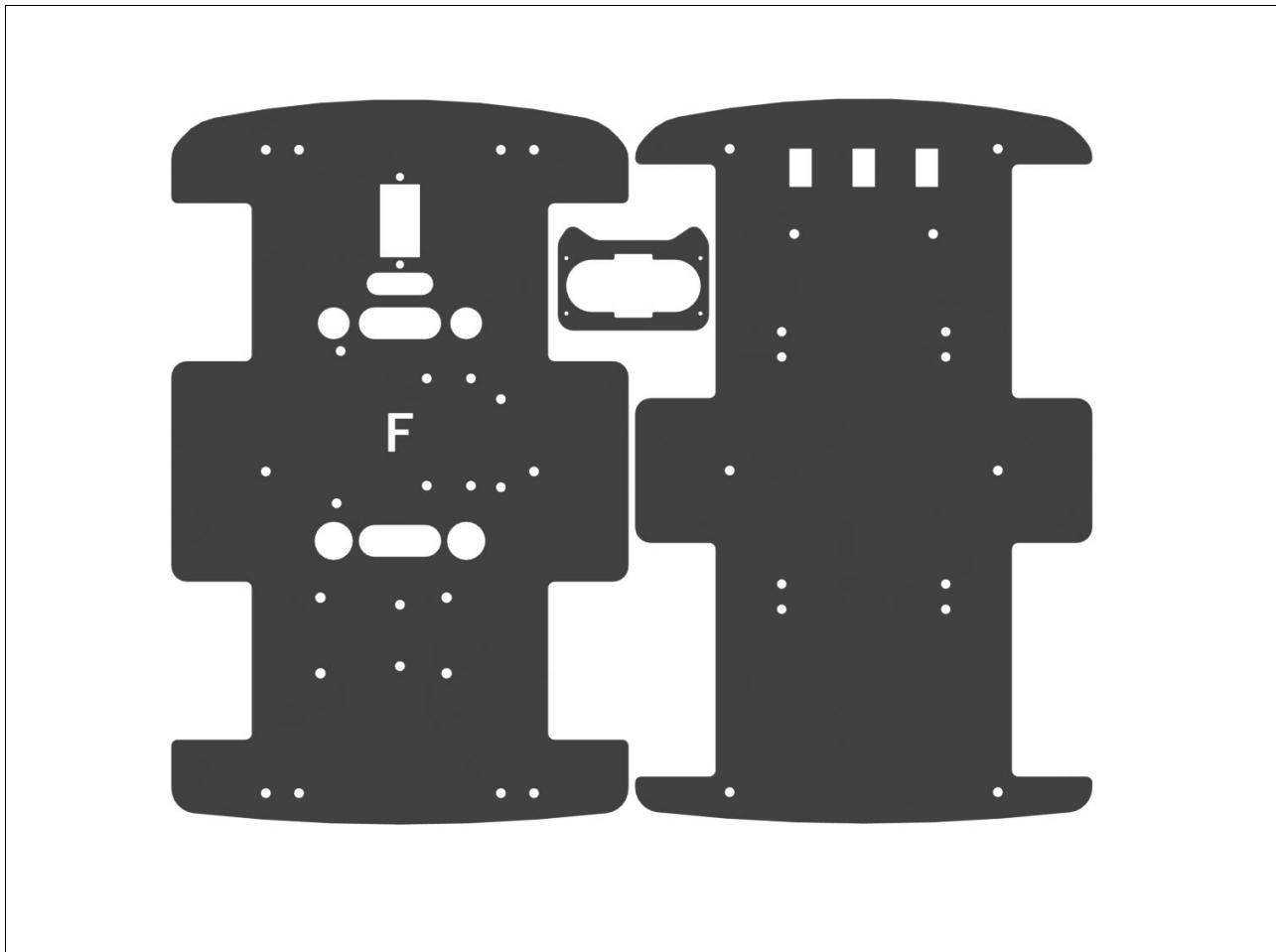
Arduino® est une marque commerciale d'Arduino LLC (<https://www.arduino.cc/>).

Contenu

Commencer.....	JE
Obtenir de l'aide et offrir une contribution	I Sécurité et précautions
I Voiture et Robot pour Raspberry Pi	
..... II À propos de Freenove	
..... III Table des matières	
..... IV Liste	
..... 1	
Préface	sept
Carte de contrôle et logiciel	8
Chapitre 0 Préparation, assemblage et jeu	11
0.1 Préparation	11
0.2 Assemblage	16
0.3 Comment jouer	26
Chapitre 1 Contrôle des composants de base	32
1.1 Moteur	32
1.2 Buzzer et niveau de batterie	41
1.3 RVB	47
1.4 Intégrer les fonctions	52
Chapitre 2 Évitement d'obstacles	61
2.1 Servo	61
2.2 Module de télémétrie ultrasonique	63
2.3 Voiture d'évitement d'obstacles automatique	68
Chapitre 3 Suivi de ligne	81
3.1 Capteur de suivi de ligne	81
3.2 Voiture de suivi de ligne	84
Chapitre 4 Commande IR	88
4.1 Télécommande et récepteur infrarouges	88
4.2 Voiture à distance IR	91
4.3 Voiture télécommandée infrarouge multifonctionnelle	94
Chapitre 5 Télécommande RF	102
5.1 Télécommande	102
5.2 Recevoir les données de la télécommande RF	105
5.3 Voiture à distance RF	110
5.4 Voiture télécommandée multifonctionnelle RF24	114
Chapitre 6 Contrôle Bluetooth	128
6.1 Configurer Bluetooth et recevoir des données	128
6.2 Voiture à distance Bluetooth	138
6.3 Voiture télécommandée Bluetooth multifonctionnelle	145
Quelle est la prochaine?.....	156

liste

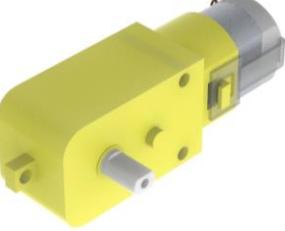
Pièces en acrylique



La surface des pièces acryliques est recouverte d'une couche de film protecteur. Vous devez d'abord le supprimer. Certains trous dans les pièces en acrylique peuvent contenir des résidus. Vous devez également les nettoyer avant de les utiliser.

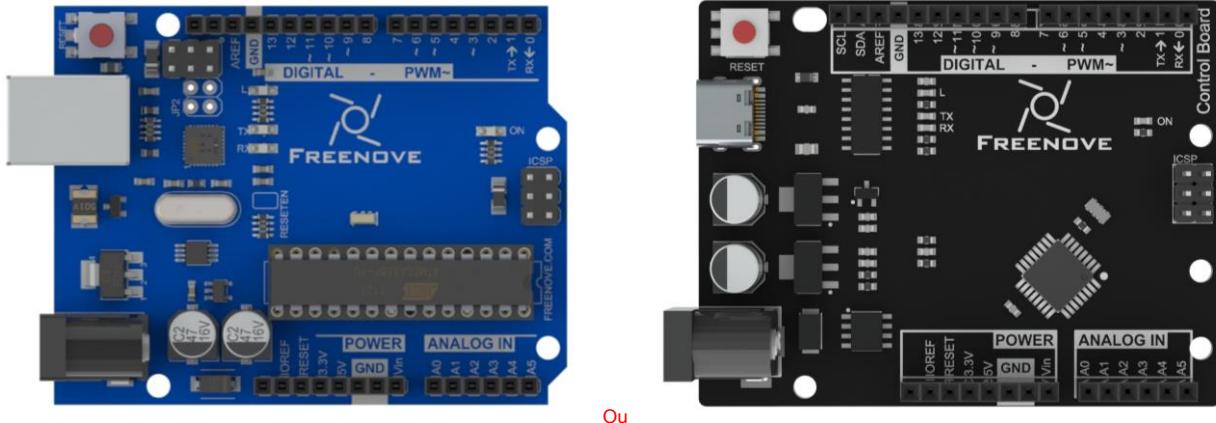
Parties mécaniques

 M3*40 Copper Standoff x6 Freenove	 M3*10 Copper Standoff x5 Freenove	 M3*10 Countersunk Head Screw x5 Freenove
 M2*10 Screw x3 Freenove	 M3*8 Screw x12 Freenove	 M3*6 Screw x22 Freenove
 M2 Nut x3 Freenove	 M3 Nut x16 Freenove	 M1.4*4 Self-tapping Screw x5 Freenove

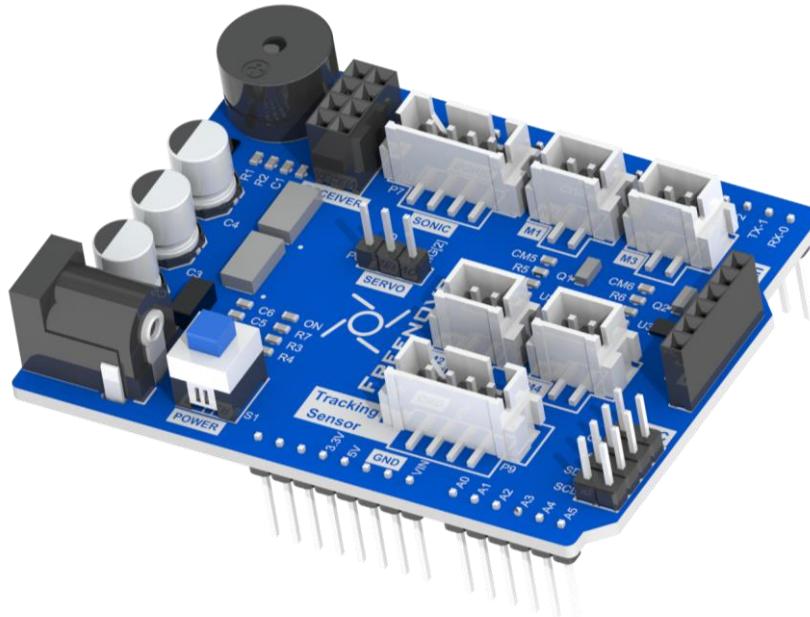
Servo x1 	Moteur x4 
Roue du conducteur x 4 	Paquet de support de moteur x 4 Vis M3 * 8, Écrou M3, Vis M3 * 30 

Parties électroniques

Carte de contrôle Freenove (Votre kit contiendra au hasard **une carte de contrôle**)



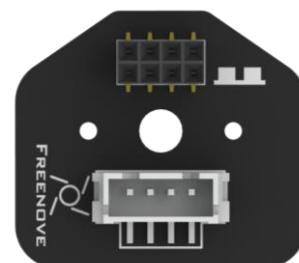
Carte d'extension Freenove 4WD

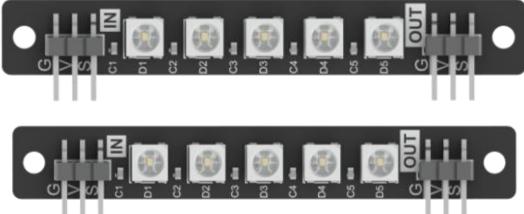
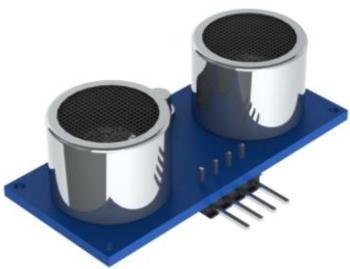
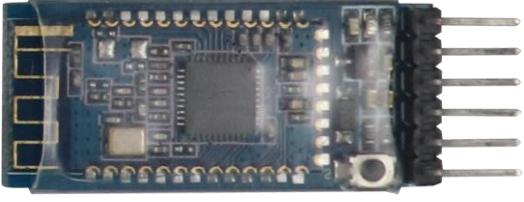
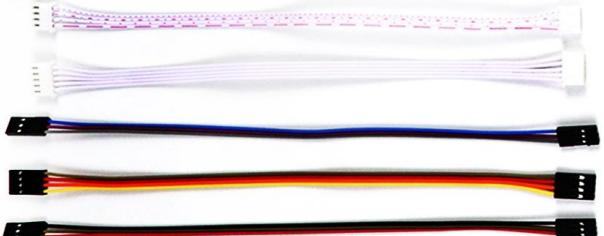
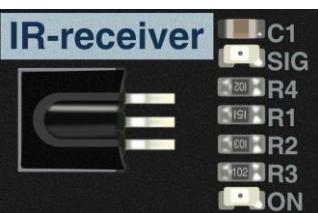
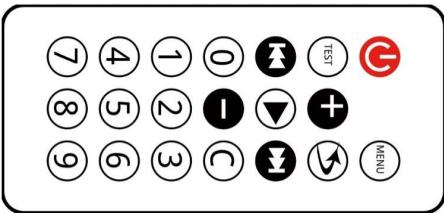


Capteur infrarouge à suivi de ligne



Connecteur de module ultrasonique



<p>WS2812B_LED_controller</p> 	<p>WS2812B_LED</p> 
<p>Module ultrasonique</p> 	<p>Bluetooth</p> 
<p>Fils</p> 	<p>Cable USB</p> 
<p>IRreceiver</p> 	<p>Ruban</p> 
<p>Télécommande IR</p> 	<p>Support de batterie</p> 

Paquet de kit de télécommande RF (**contenu uniquement dans la version avec télécommande RF**)

Le kit à distance a également deux versions, **noir** ou **bleu**. Vous en recevrez un au hasard.



Outils

Tournevis cruciforme x1



Prise x1 (**un seul est inclus dans le**



Nécessaire mais PAS inclus

18650 3,7 V **rechargeable** batterie au lithium x2

Il est plus facile de trouver la bonne batterie sur eBay que Amazon.

Prière de se référer à **About_Battery.pdf** dans le dossier décompressé.



Préface

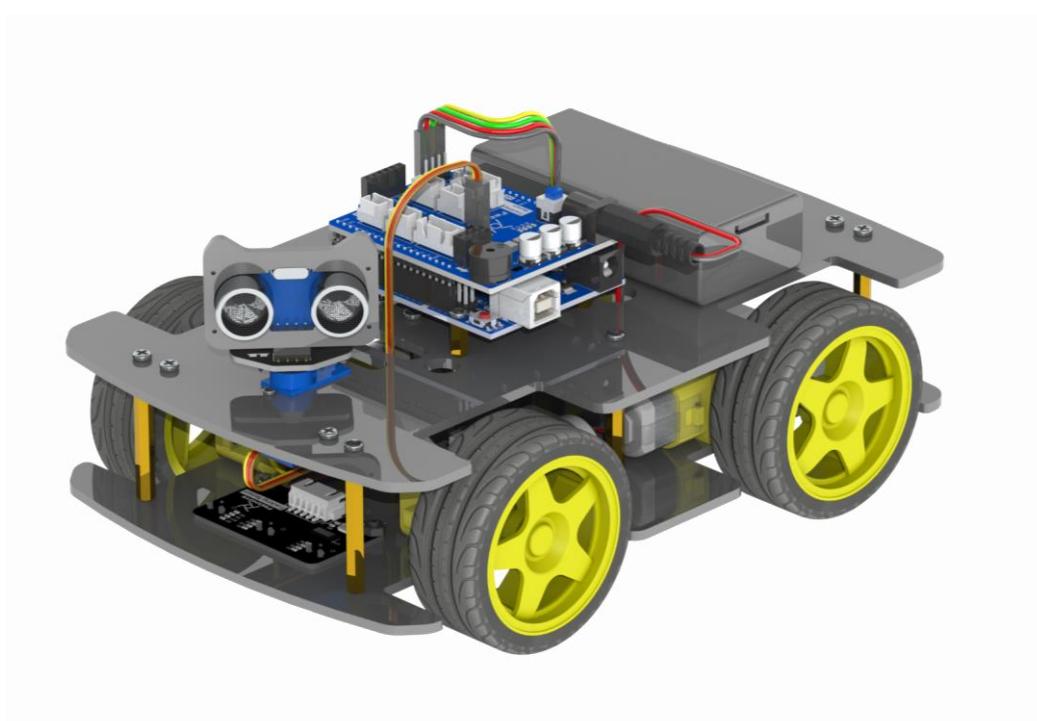
Cette voiture Smart 4WD est basée sur Arduino.

Ce tutoriel commencera par le contrôle des composants électroniques tels que la LED, le moteur, le servo, à la construction d'une voiture multifonctionnelle.

Veuillez suivre le tutoriel étape par étape avec patience.

Si vous avez des inquiétudes, n'hésitez pas à nous contacter. Notre équipe d'assistance fournira un service client rapide et gratuit.

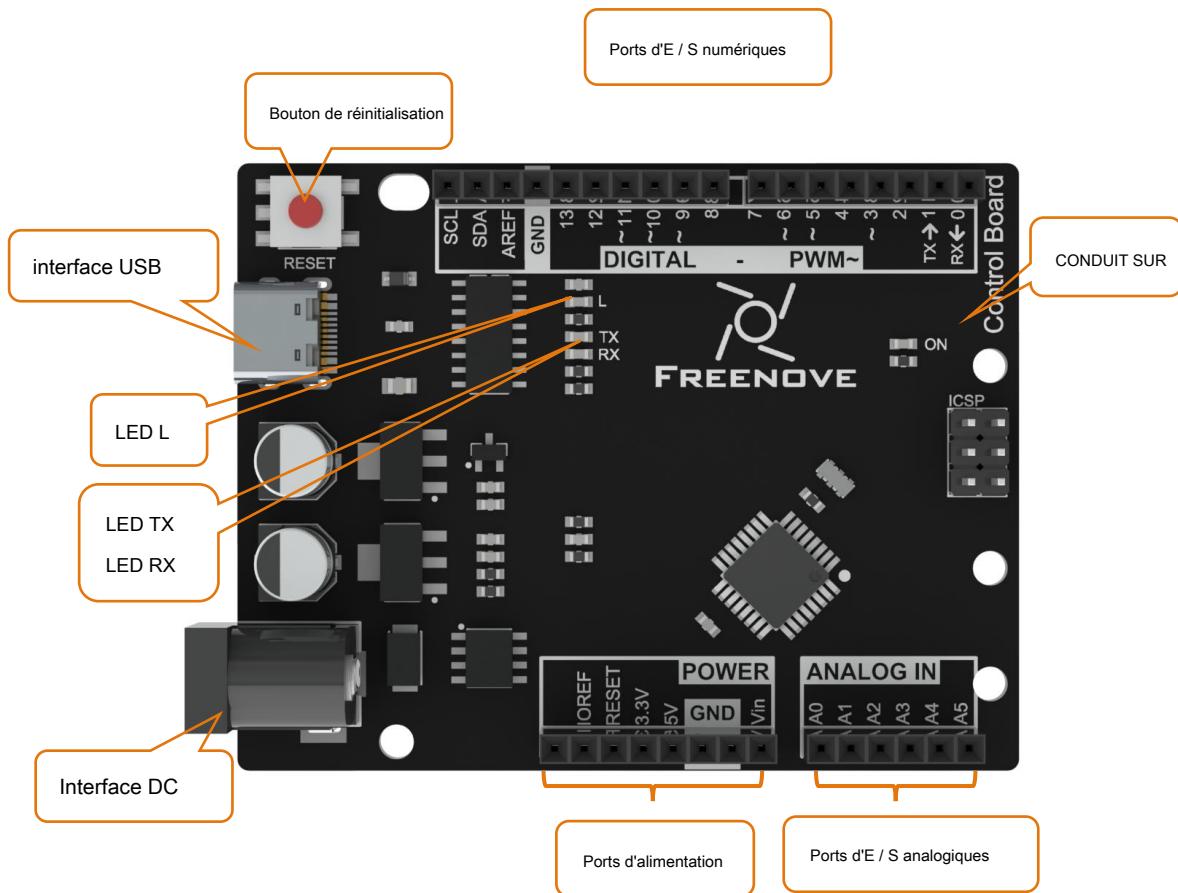
support@freenove.com



Carte de contrôle et logiciel

Tableau de contrôle

Le schéma du tableau de commande Freenove est illustré ci-dessous:

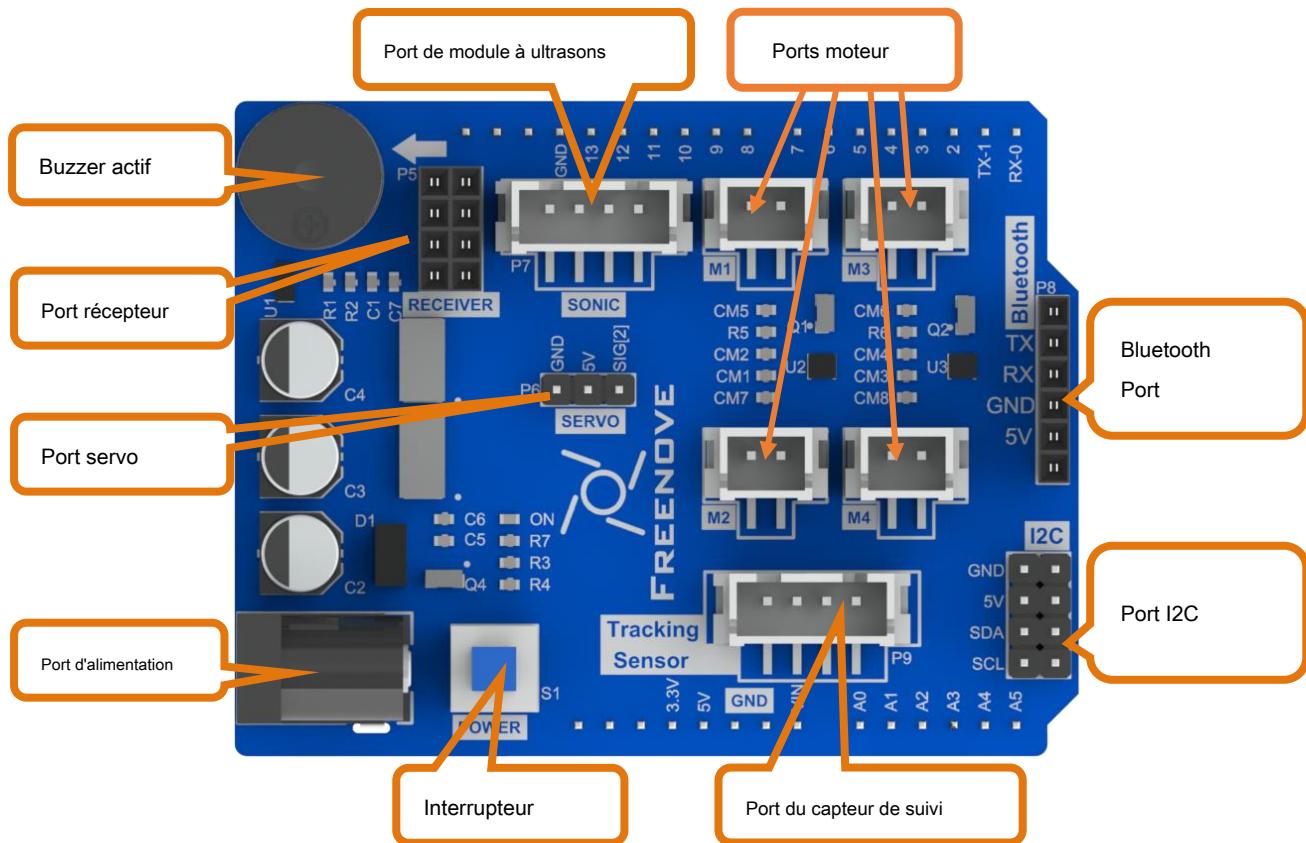


- Les ports d'E / S numériques sont utilisés pour se connecter à d'autres composants ou modules, pour recevoir un signal d'entrée ou pour envoyer un signal de commande. Habituellement, nous le nommons en ajoutant un «D» devant le nombre, tel que D13. Seuls les ports d'E / S numériques avec «~» peuvent sortir PWM, broches 3, 5, 6, 9, 10, 11.
- L'interface USB est utilisée pour fournir l'alimentation, télécharger le code ou communiquer avec le PC. La LED L est connectée au port E / S numérique 13 (D13).
- La LED TX, RX est utilisée pour indiquer l'état de la communication série.
- L'interface CC est connectée à l'alimentation CC pour alimenter la carte. Les ports d'alimentation peuvent alimenter les composants et modules électroniques. Les ports d'E / S analogiques peuvent être utilisés pour mesurer les signaux analogiques.
- La LED ON est utilisée pour indiquer l'état de l'alimentation.

Si vous souhaitez en savoir plus sur l'utilisation du panneau de contrôle Freenove pour créer des projets intéressants, veuillez visiter <http://www.freenove.com/store.html> pour les kits Arduino conçus pour les débutants.

Carte d'extension

La carte d'extension Freenove 4WD est ci-dessous:



Le port du récepteur peut se connecter au module RF ou au module IR.

La relation entre la carte d'extension et la carte de commande est la suivante.

Broches des ports Arduino	du multiplexage des broches d'extension	Description	
0	UART-Bluetooth		Bluetooth
1			
2	Servo		
3	Direction-droite		Sens du moteur droit Sens du
4	Direction-gauche		moteur gauche
5	Moteur-droit		Vitesse du moteur droit -PWM Vitesse
6	Moteur-gauche		du moteur droit -PWM
sept	Trigonométrie		Module ultrasonique
8	Écho		
9	SPI-NRF24L01	Télécommande IR	Télécommande CE / IR
dix			CS
11	SPI-NRF24L01		MOSI
12			MISO
13			SCK
A0	Voltage de batterie	Avertisseur sonore	L'ADC utilise une référence interne de 5 V et la tension d'acquisition est la tension de la batterie * 1/4. Sortie vers le comparateur Buzzer. à travers le Quand plus haut que le 4.5V, sortie HIGH.
A1	Capteur de suivi de ligne		La gauche
A2			Milieu
A3			Droite
A4	I2C		SDA
A5			SCL

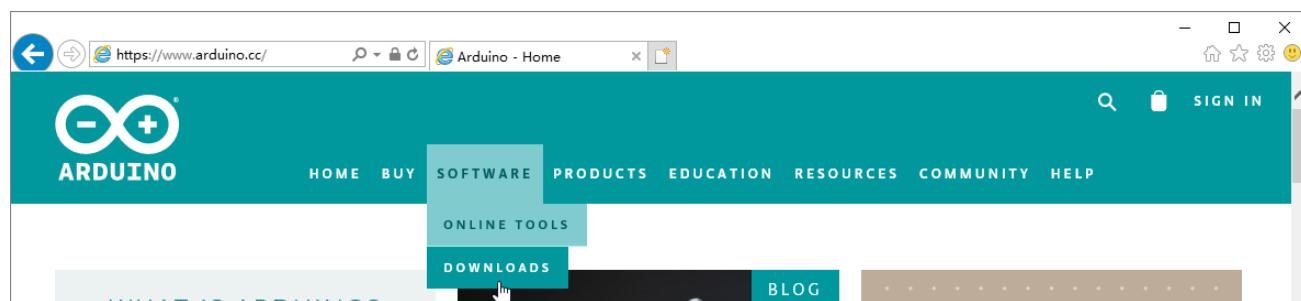
Chapitre 0 Préparation, assemblage et jeu

Si vous avez des préoccupations, n'hésitez pas à nous contacter via support@freenove.com

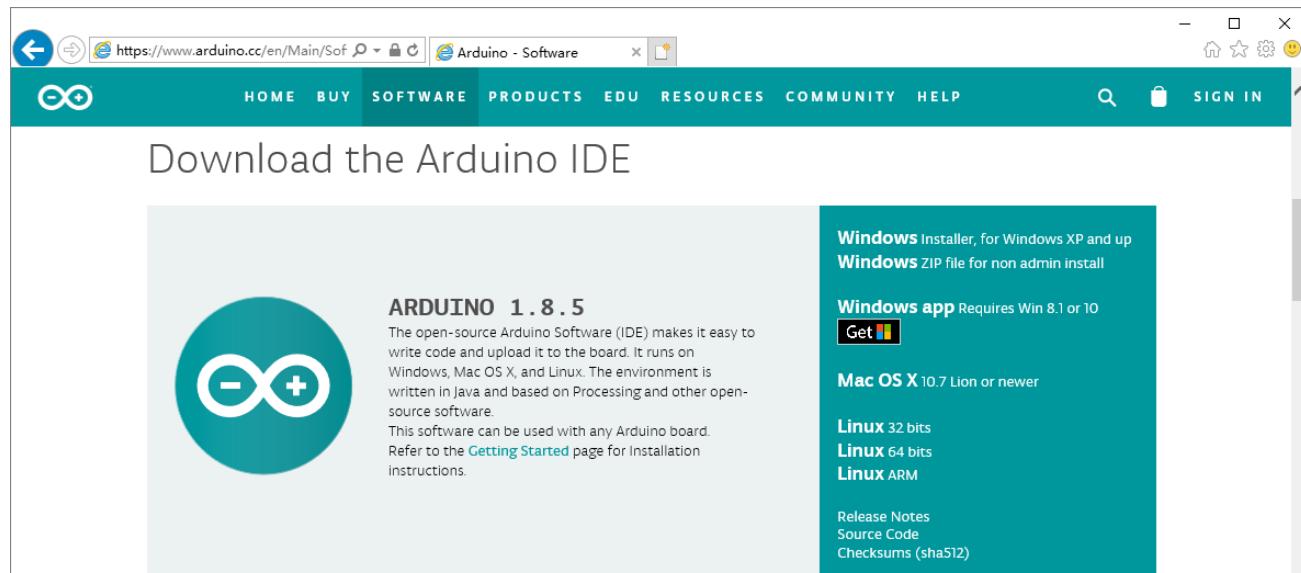
0.1 Préparation

Étape 1 Téléchargez et installez IDE.

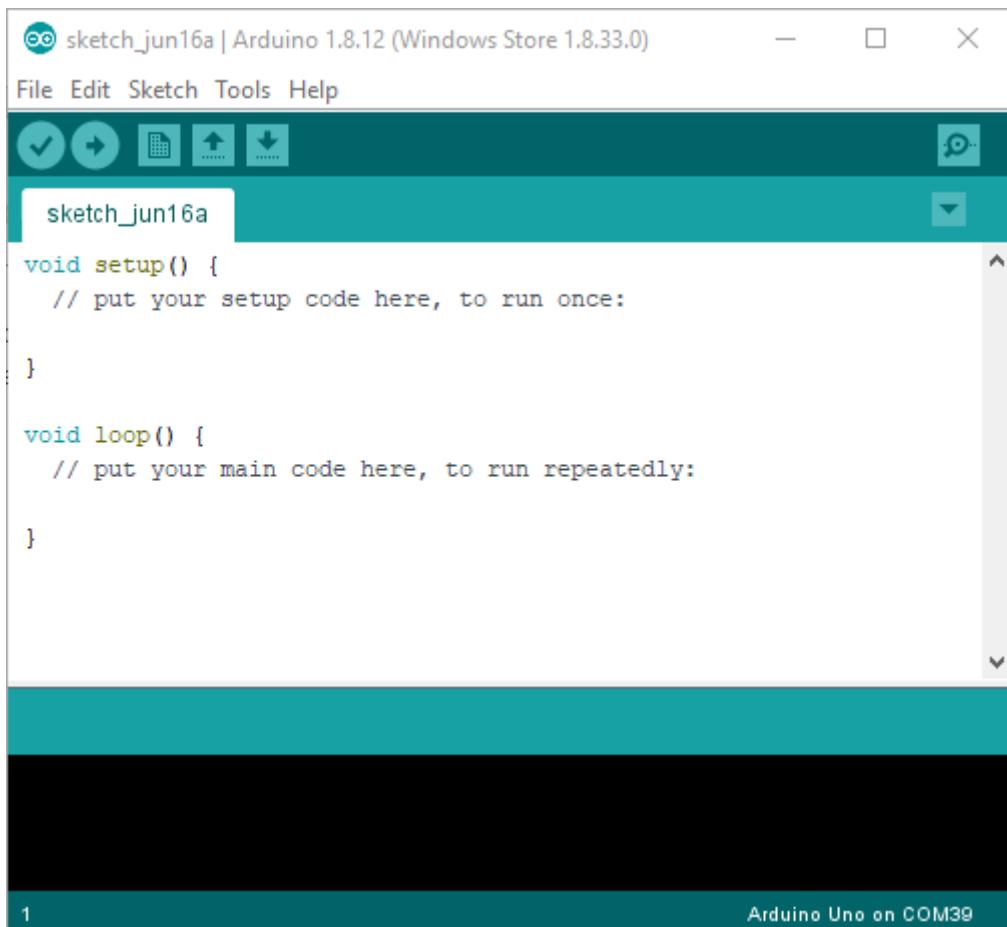
Veuillez visiter <https://www.arduino.cc>, puis cliquez sur «LOGICIEL» - «TÉLÉCHARGEMENTS» pour accéder à la page de téléchargement.



Trouvez "Téléchargez l'IDE Arduino". Pour les utilisateurs de Microsoft Windows, veuillez cliquer sur "Windows Installer".



Une fois le téléchargement terminé, exécutez le programme d'installation et terminez l'installation. Ouvrez le logiciel Arduino, l'interface du logiciel Arduino est la suivante:



Les programmes écrits avec le logiciel Arduino (IDE) sont appelés **croquis**. Ces esquisses sont écrites dans l'éditeur de texte et sont enregistrées avec l'extension de fichier. **ino**. L'éditeur a des fonctionnalités pour couper / coller et pour rechercher / remplacer du texte. La zone de message donne des informations lors de l'enregistrement et de l'exportation et affiche également les erreurs. La console affiche la sortie de texte par le logiciel Arduino (IDE), y compris les messages d'erreur complets et d'autres informations. Le coin inférieur droit de la fenêtre affiche la carte configurée et le port série. Les boutons de la barre d'outils vous permettent de vérifier et de télécharger des programmes, de créer, d'ouvrir et d'enregistrer des esquisses et d'ouvrir le moniteur série.



- Vérifier, vérifie votre code pour les erreurs de compilation.
- Téléchargez, compile votre code et le télécharge sur la carte configurée. Nouveau, crée une nouvelle esquisse.
- Ouvrir, présente un menu de tous les croquis de votre carnet de croquis. En cliquant dessus, vous l'ouvrirez dans la fenêtre actuelle en écrasant son contenu.
- Enregistrer, enregistre votre croquis.
- Serial Monitor, ouvre le moniteur série.

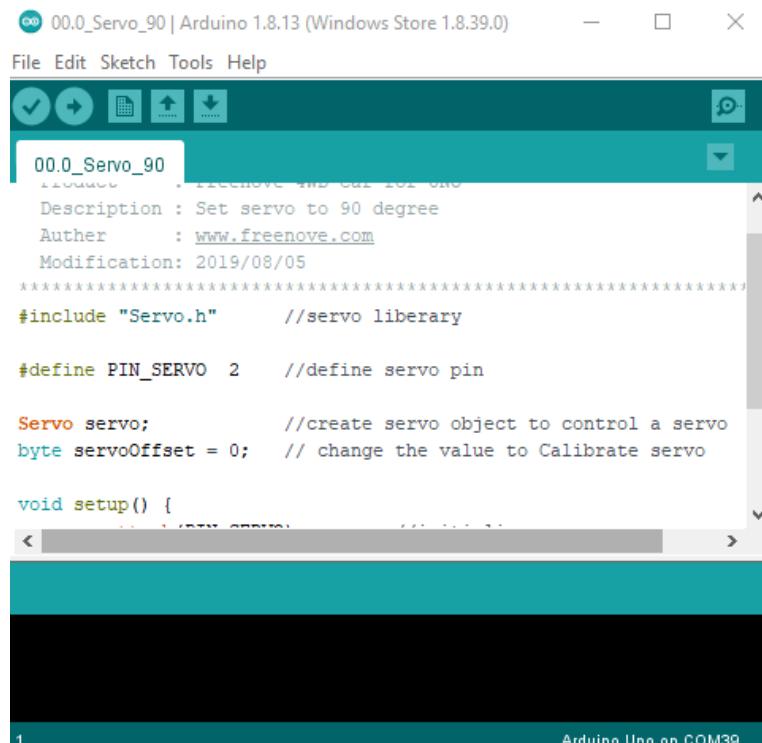
Des commandes supplémentaires se trouvent dans les cinq menus: Fichier, Edition, Esquisse, Outils, Aide. Les menus sont contextuels, ce qui signifie que seuls les éléments pertinents pour le travail en cours sont disponibles.

Étape 2 Télécharger le code servo (nécessaire)

Télécharger le code dans **Freenove 4WD Car Kit\Sketches\00.0 Servo 90** au tableau de commande Freenove de la voiture.

Ce code sera utilisé dans l'étape d'assemblage. Ne passez pas cette étape. Veuillez supprimer le module BluetoothModule lorsque vous téléchargez le code.

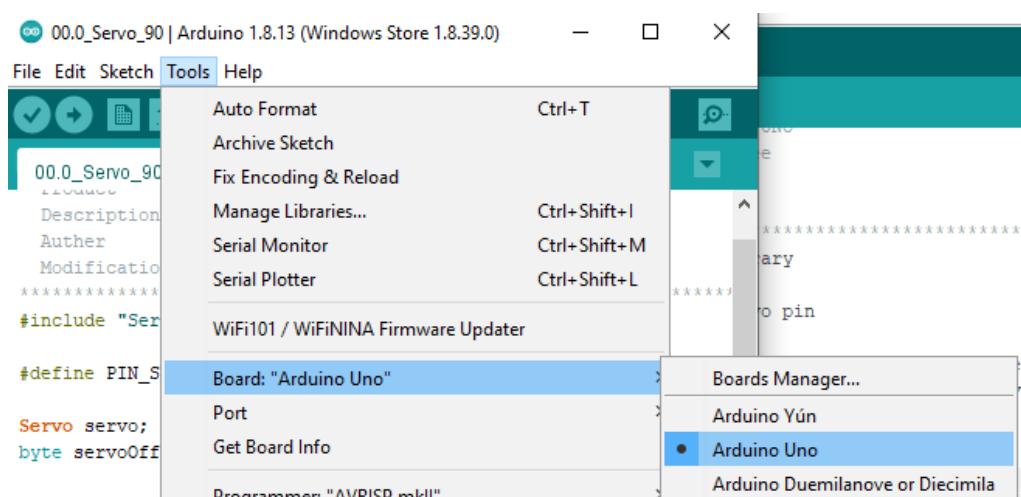
Ouvrez l'exemple d'esquisse " **00.0_Servo_90** " avec Arduino IDE.



Si le port ne fonctionne pas, veuillez installer le pilote de la carte.

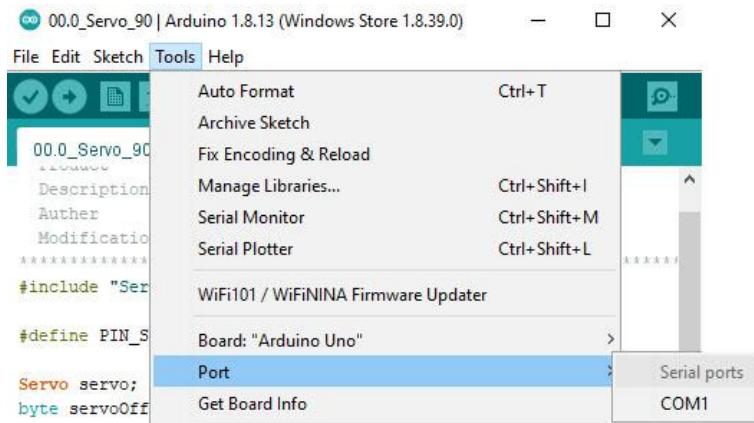
Freenove 4WD Car Kit \ Drivers

Sélectionnez le tableau " Arduino / Genuino Uno ".

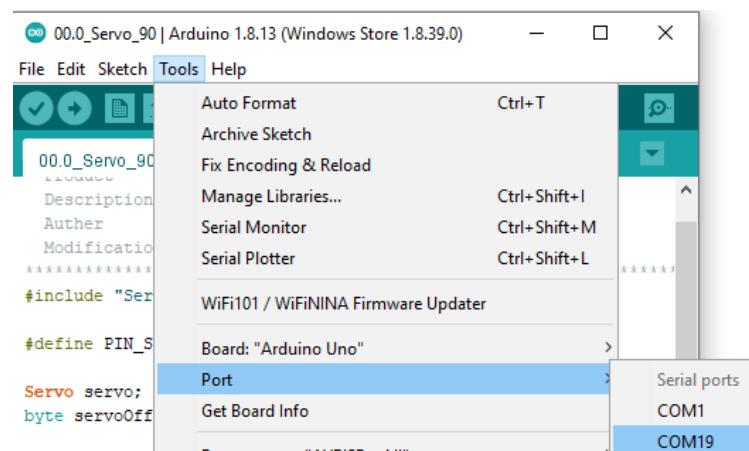


Vérifiez le port avant de connecter la carte de contrôle à l'ordinateur.

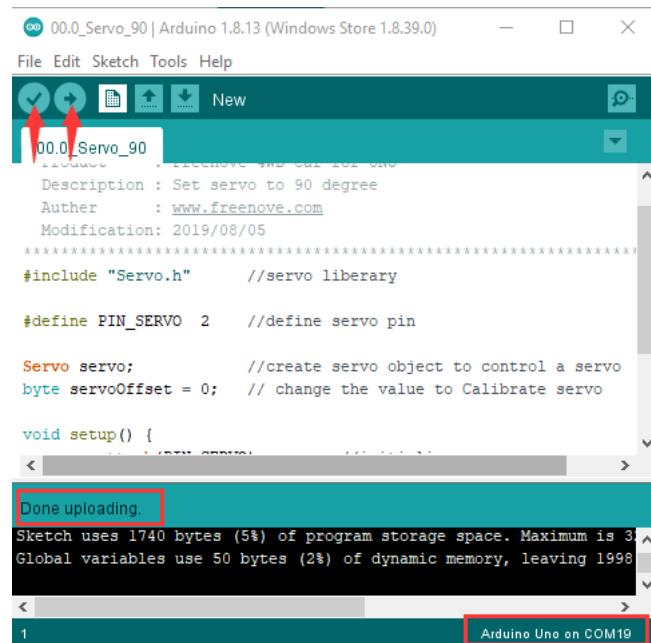
Remarque Votre port peut être différent de la figure suivante. Sur **Les fenêtres**: Cela peut être COM4, COM5 (Arduino Uno) ou quelque chose comme ça. Sur **Mac**: Il peut s'agir de / dev / cu. **usbserial-710**, /dev/cu.usbmodem7101 (Arduino Uno) ou quelque chose comme ça. Sur **Linux**: Cela peut être / dev / ttyUSB0, / dev / ttyACM0 ou quelque chose comme ça.



Connectez la carte de contrôle à l'ordinateur via un câble USB. Et puis vérifiez à nouveau le port. Vous trouverez un COM ajouté. C'est le port du panneau de contrôle. Ici, c'est COM19. Dans votre ordinateur, ce serait un numéro différent.



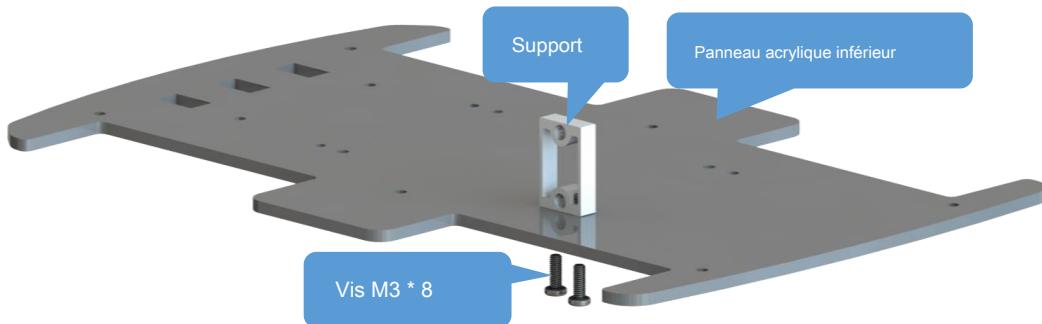
Cliquez sur le bouton «Vérifier» et sur le bouton «Télécharger». «Téléchargement terminé» indique que le code a été téléchargé avec succès.



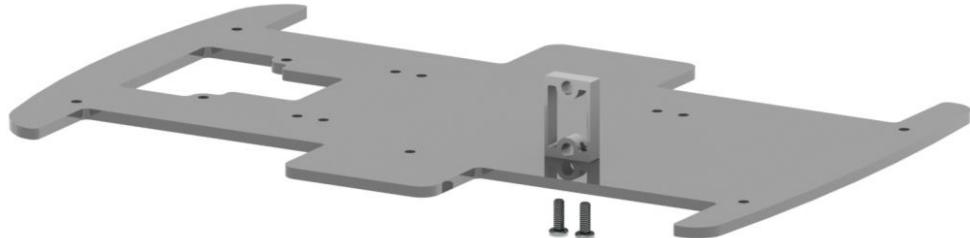
0.2 Assemblage

Si vous avez des préoccupations, n'hésitez pas à nous contacter via support@freenove.com

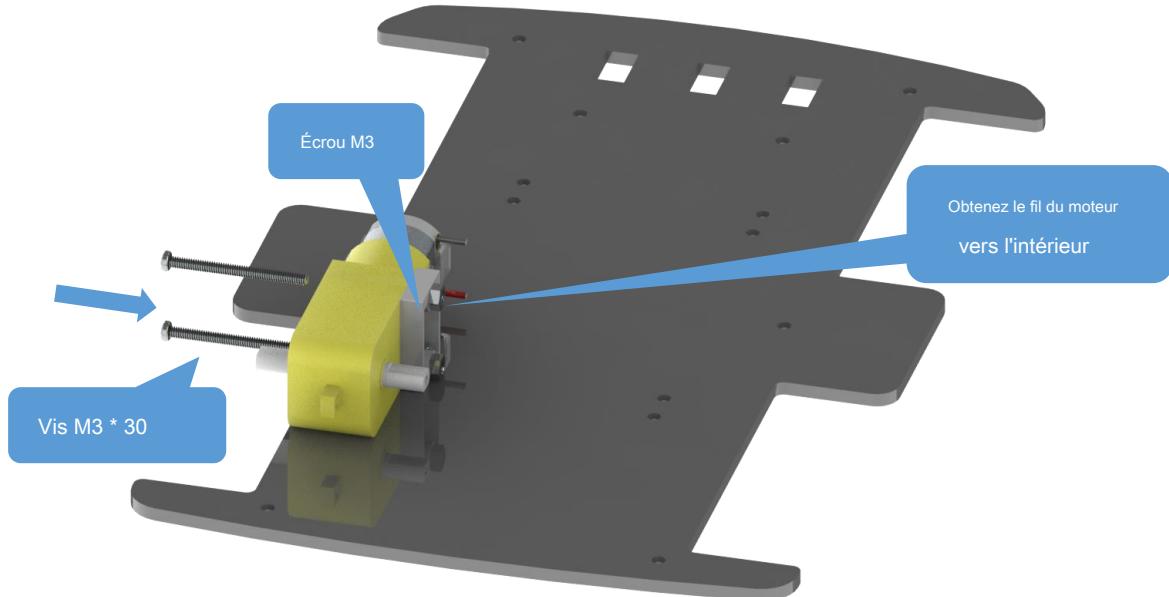
Dans ce chapitre, nous apprendrons comment assembler cette voiture. Cela prendra du temps. Veuillez être patient. Installez le support de moteur sur le panneau acrylique inférieur, avec la vis M3 * 8 dans le même sac.



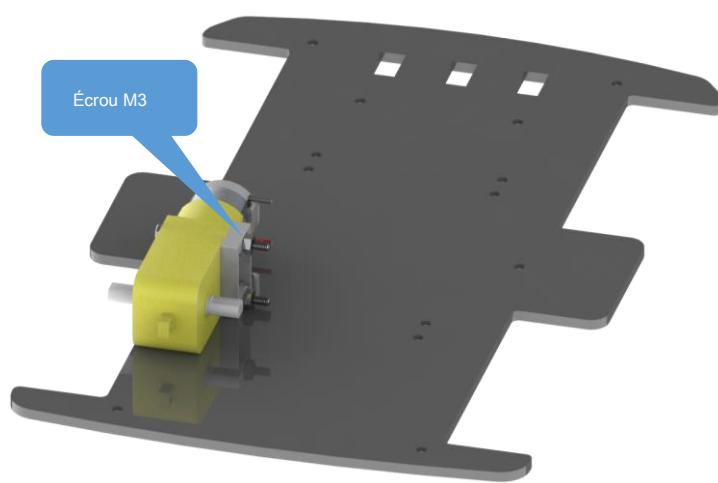
Dans votre kit, le panneau acrylique inférieur peut ressembler à ci-dessous:



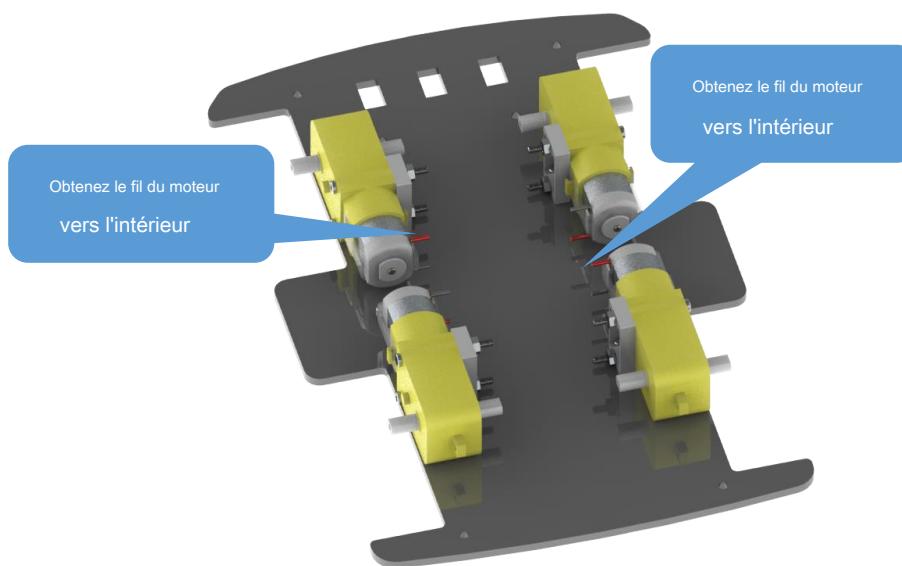
Installez le support de moteur sur le panneau acrylique inférieur, avec la vis M3 * 30 et l'écrou M3 dans le même sac.



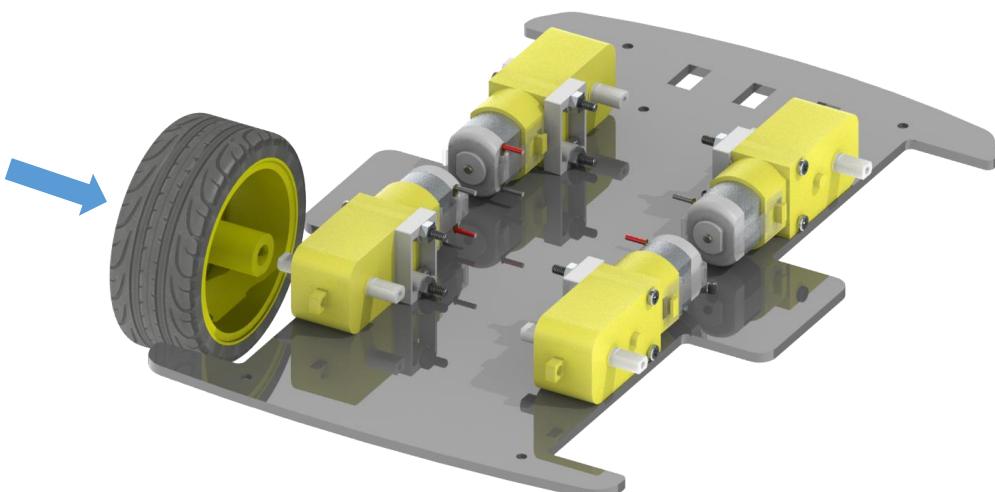
Et puis un moteur sera installé avec succès.



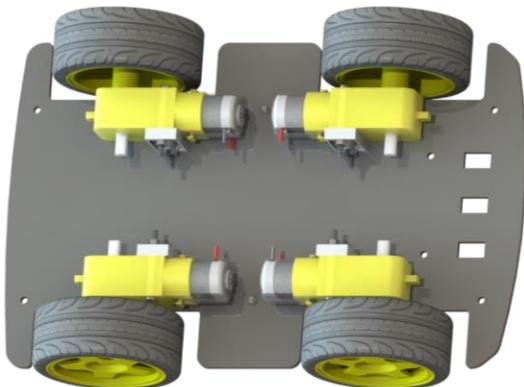
Installez les 3 autres ensembles de moteurs avec la même méthode.



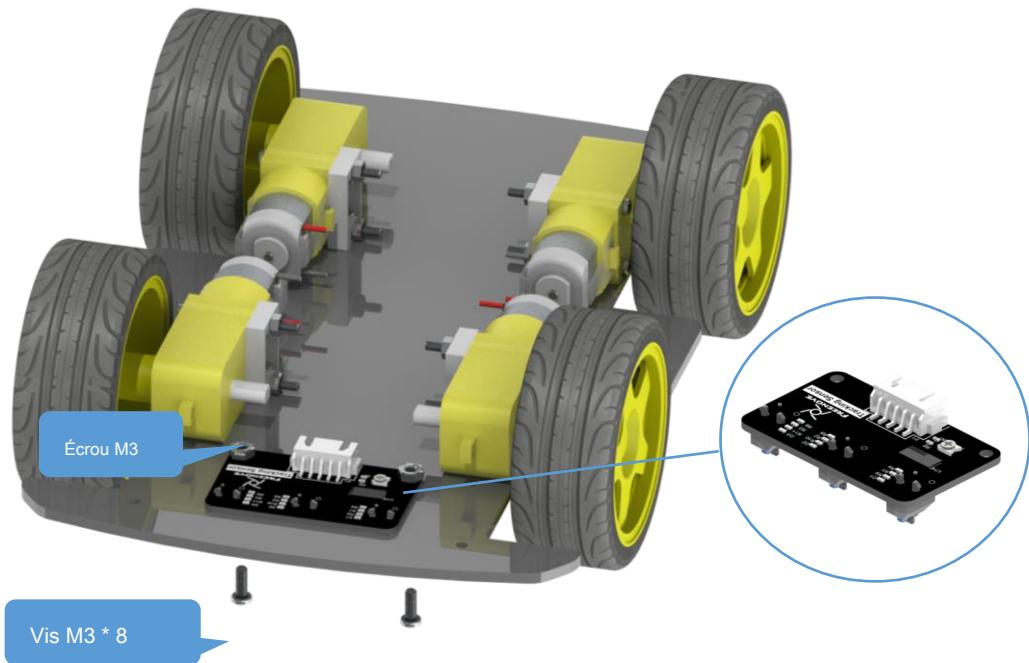
Installez la roue. Notez que le trou n'est pas rond.



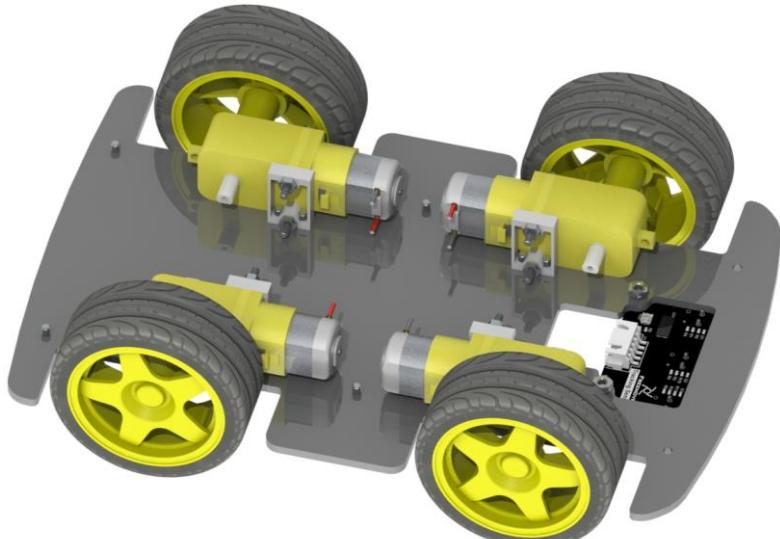
Installez les 3 autres roues.



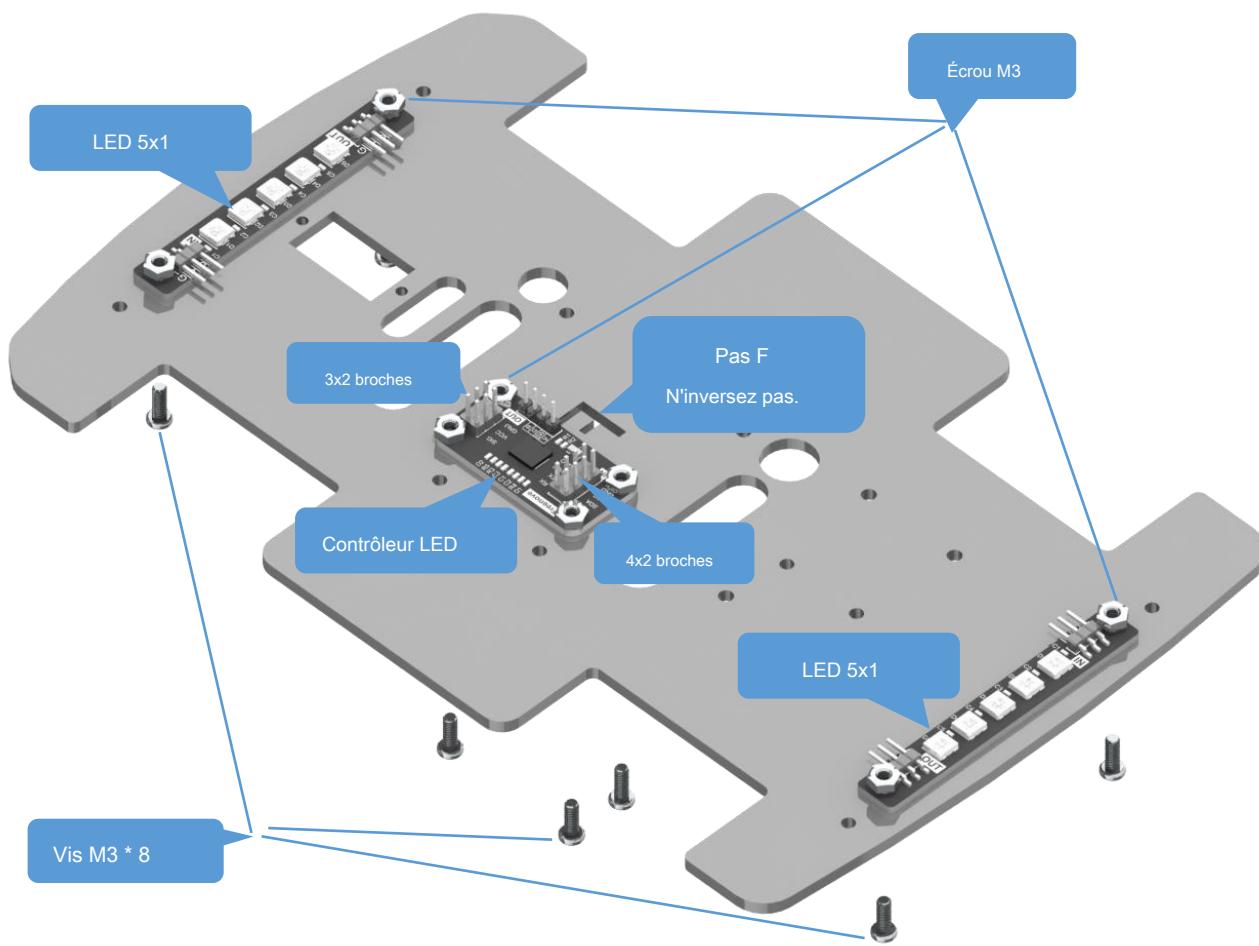
Installez le module de suivi de ligne avec ScrewM3 * 8 et Nut M3.



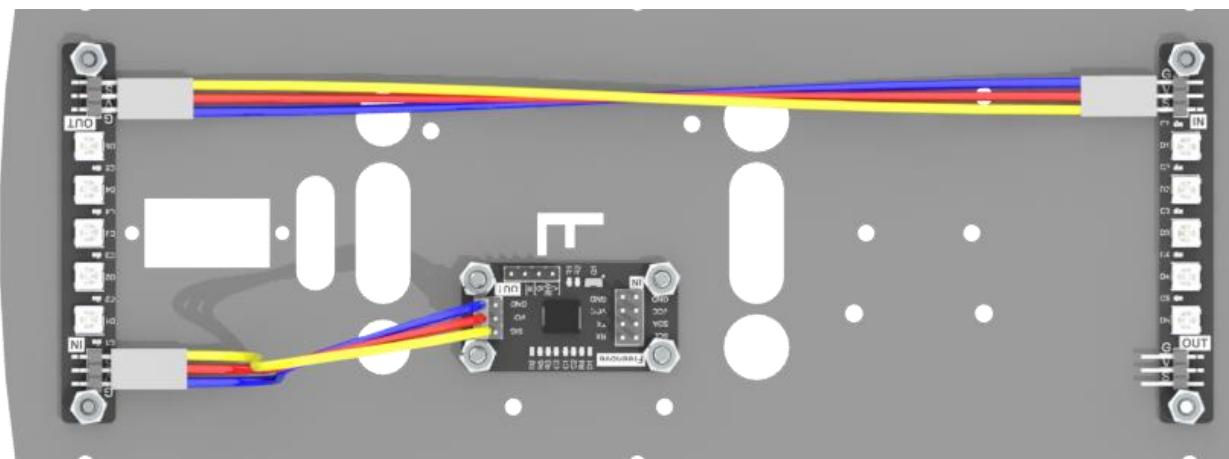
Si vous avez un autre type de panneau acrylique inférieur, veuillez installer le module de suivi comme ci-dessous.



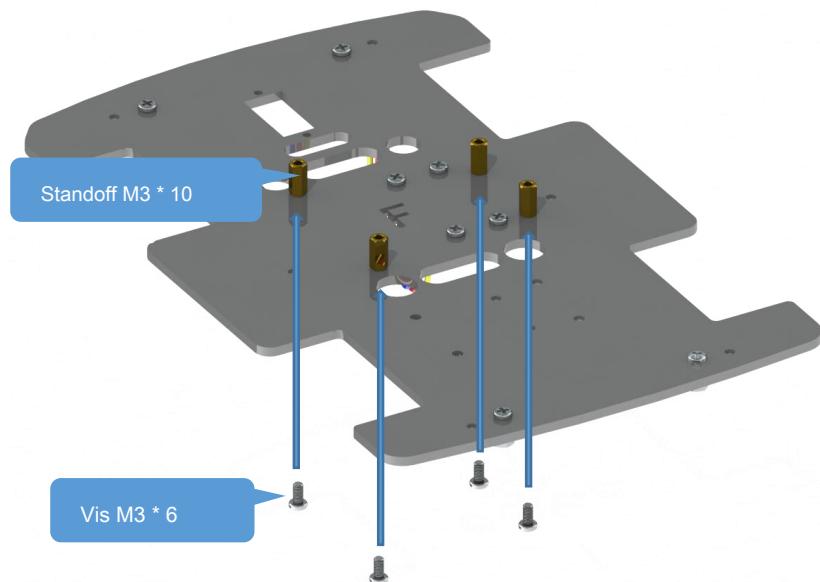
Installez le contrôleur LED 5x1 et LED sur le panneau acrylique supérieur.



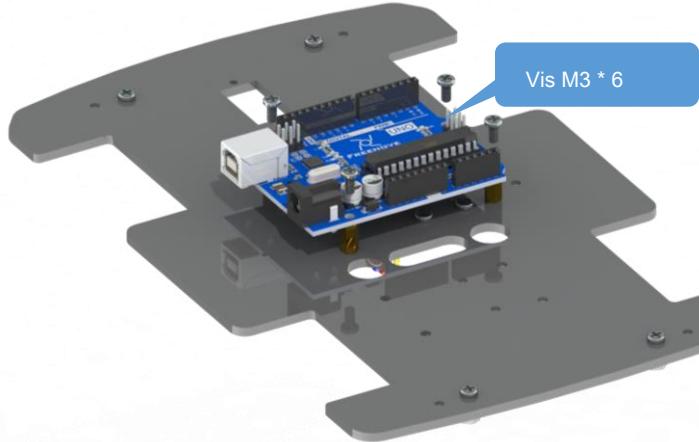
Connectez le câblage du contrôleur LED et de la LED 5X1. Le fil doit être tordu à 180 °.



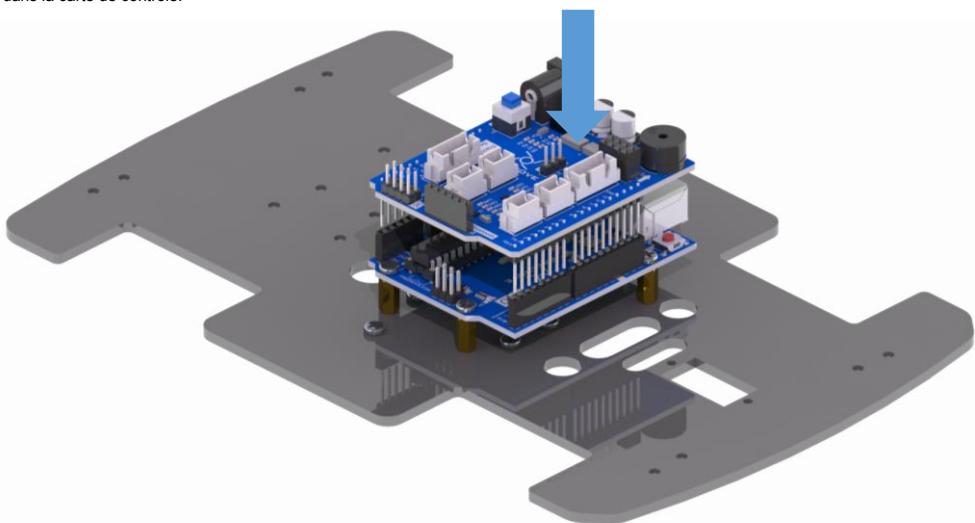
Installez le support sur le panneau acrylique supérieur.



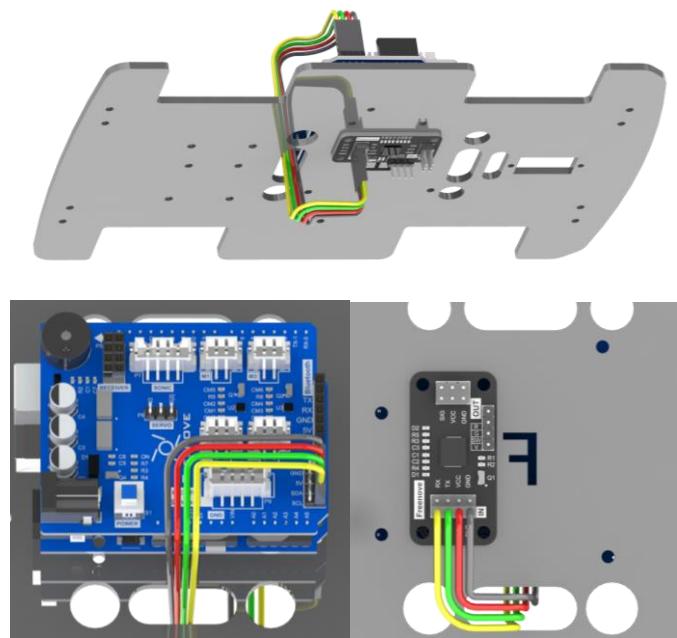
Installez la carte de contrôle Freenove.



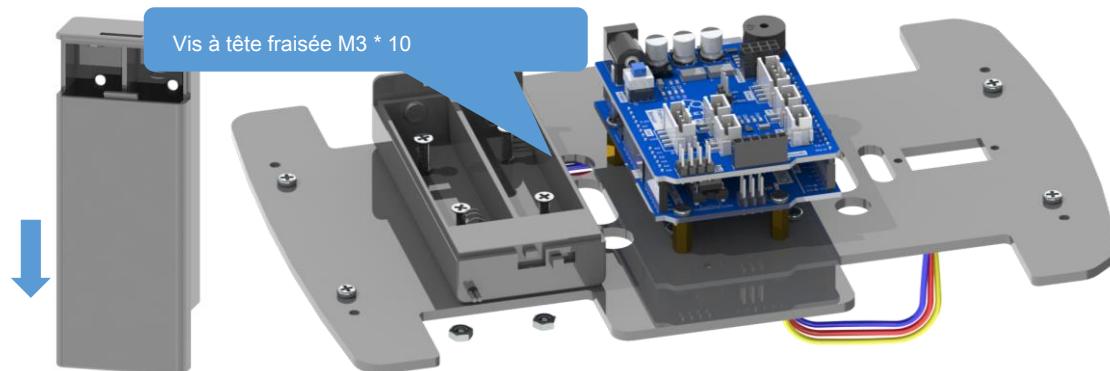
Branchez l'extension dans la carte de contrôle.



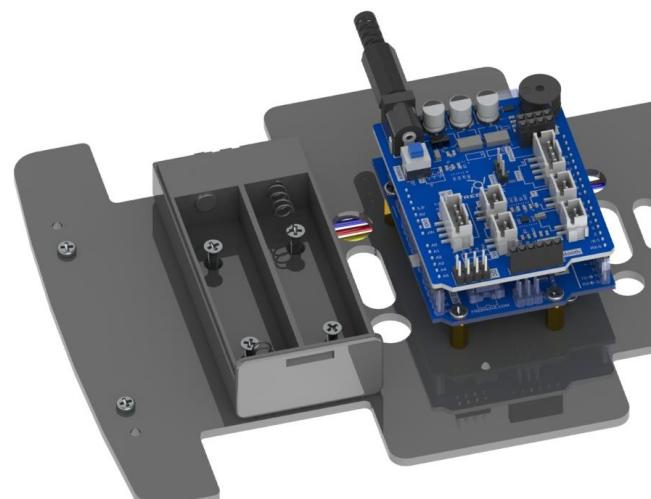
Connectez le contrôleur LED à la carte d'extension avec le fil de liaison FF 4P. Les fils de LED sont cachés. GND-GND, 5V-VCC, SCAL-RX, SDA-TX.



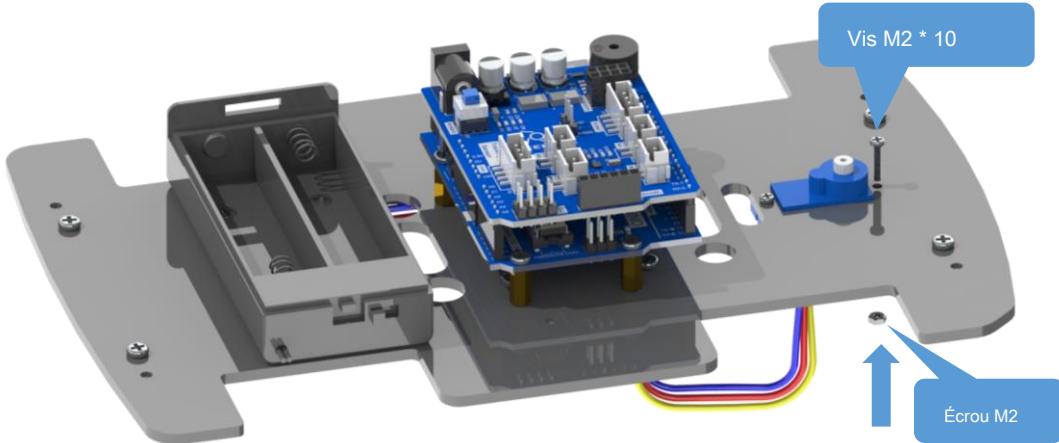
Installez le support de batterie.



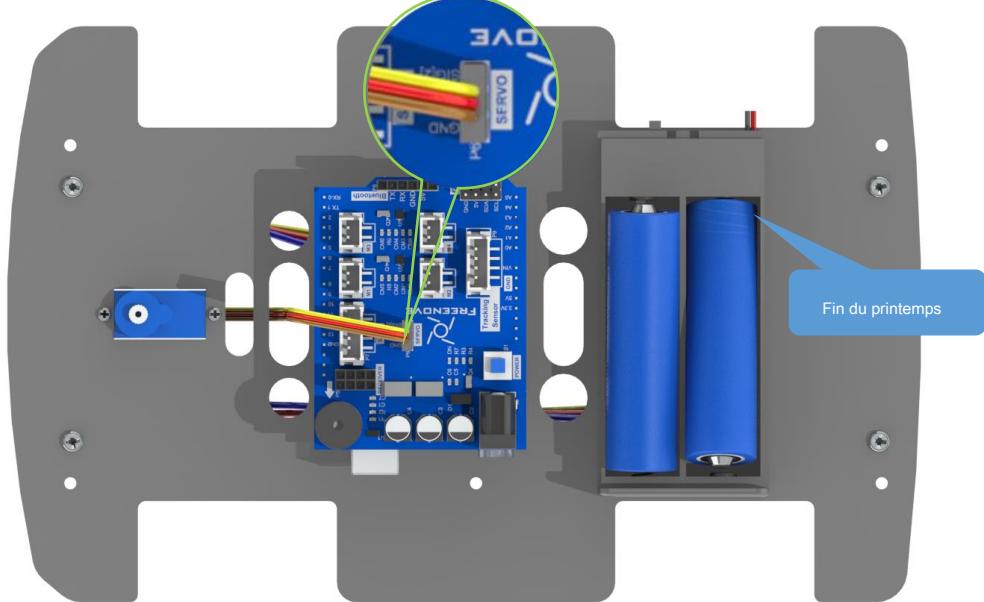
Certains supports de batterie ont un câble court. Veuillez faire pivoter le support de batterie à 180 ° pour l'installer.



Installez le servo.

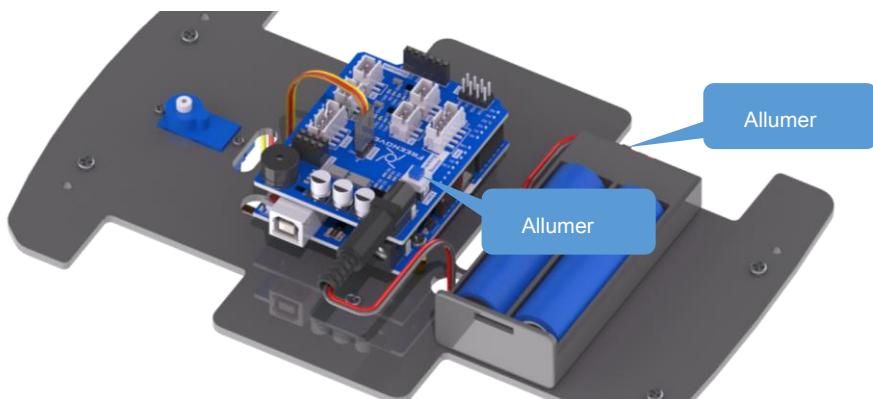


Installez la batterie. Et connectez le servo au port servo, broche jaune-SIG, rouge-5v, marron-GND.



18650 3,7 V **rechargeable** batterie au lithium x2 Il est plus facile de trouver la bonne batterie sur [eBay](#) que [Amazon](#).

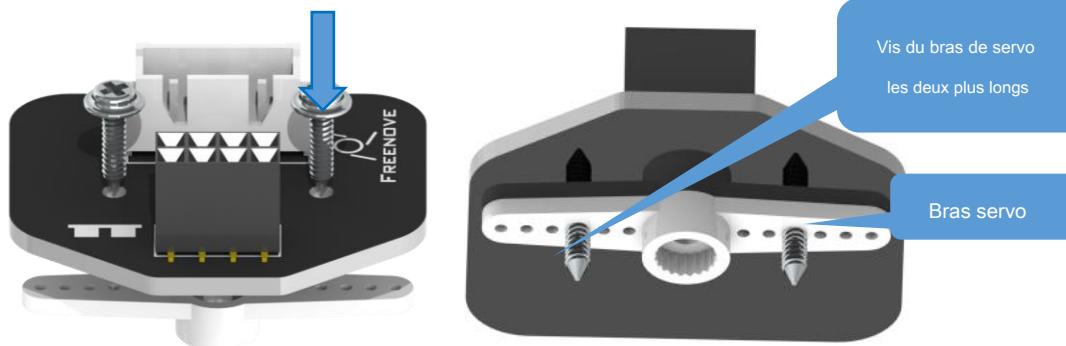
Connectez l'alimentation à la carte d'extension (la carte supérieure, pas la carte de contrôle).



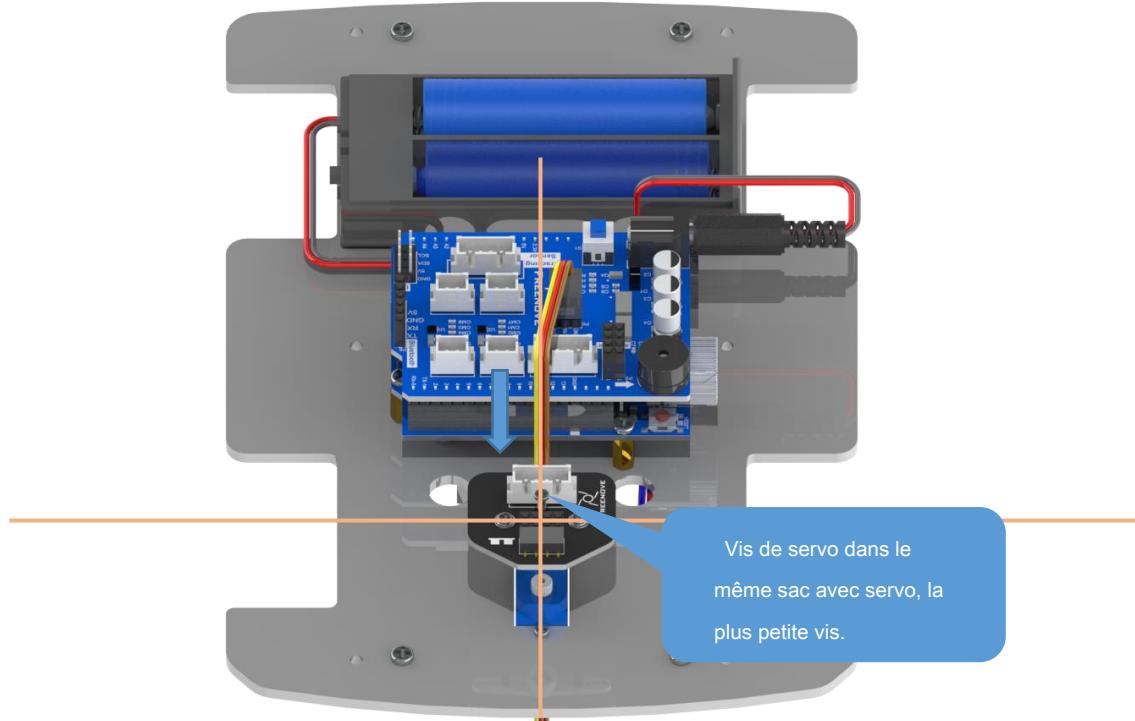
Si vous avez déjà téléchargé du code, activez simplement l'interrupteur de la carte d'extension et du support de batterie. Sinon, vous avez besoin [télécharger le code](#).

Certains supports de batterie ont un câble court. Veuillez faire pivoter le support de batterie à 180 ° pour l'installer.

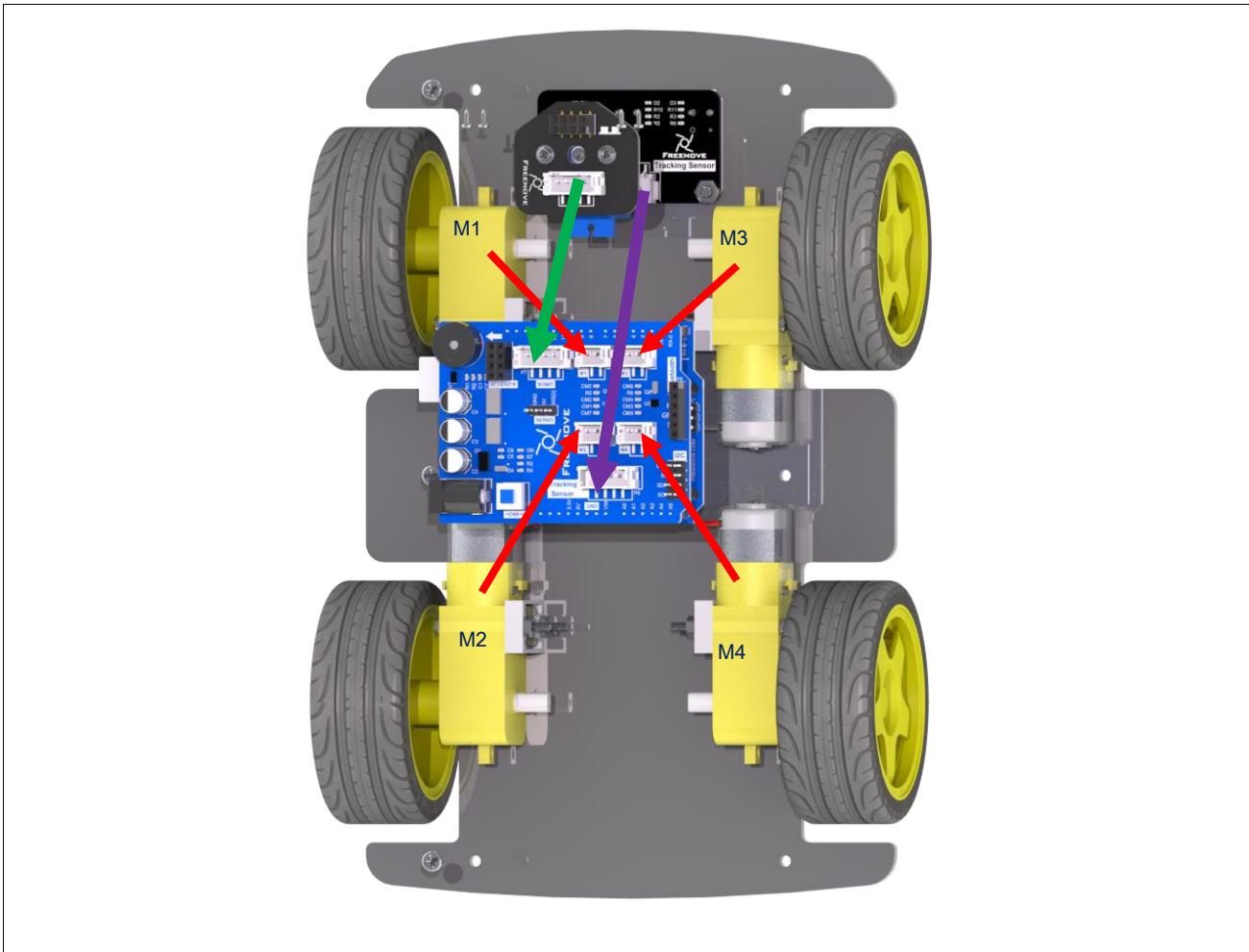
Installez le bras servo avec le connecteur du module sonique. La vis et le bras sont dans le même sac avec le servo.



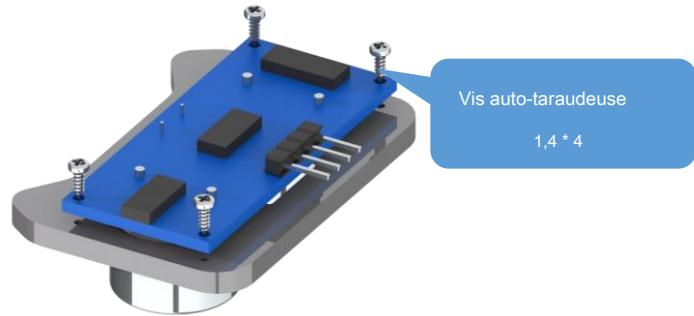
Connectez le bras servo au servo. Assurez-vous que le bras de servo est installé à 90 degrés.



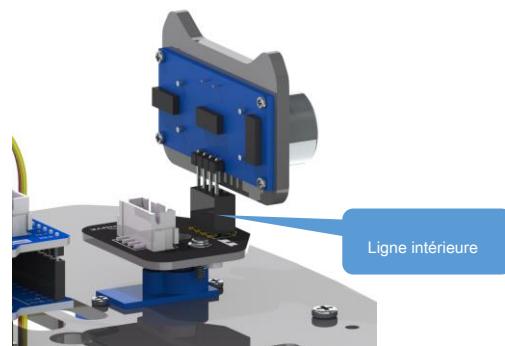
Connectez différents composants à leurs ports correspondants comme ci-dessous.



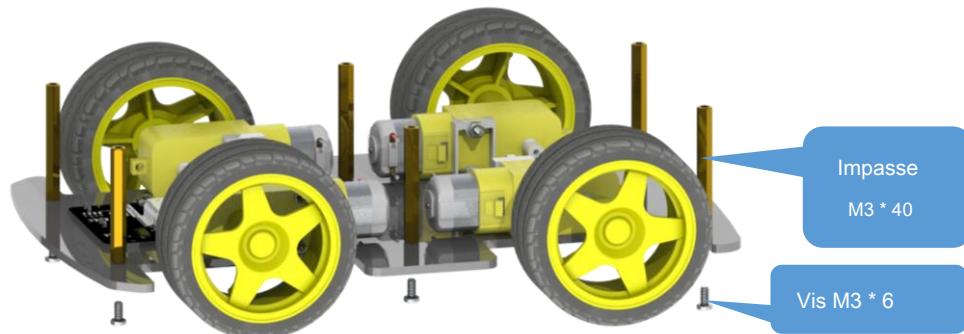
Installez le module ultrasonique sur le panneau acrylique.



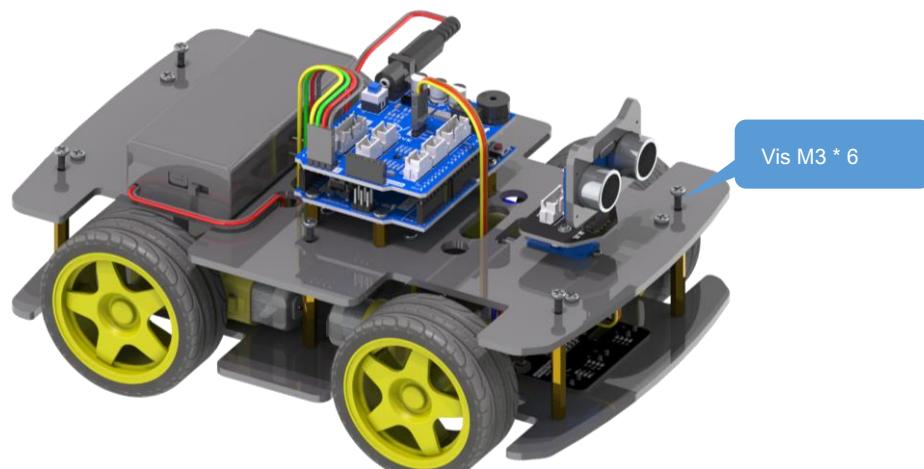
Branchez le module ultrasonique.



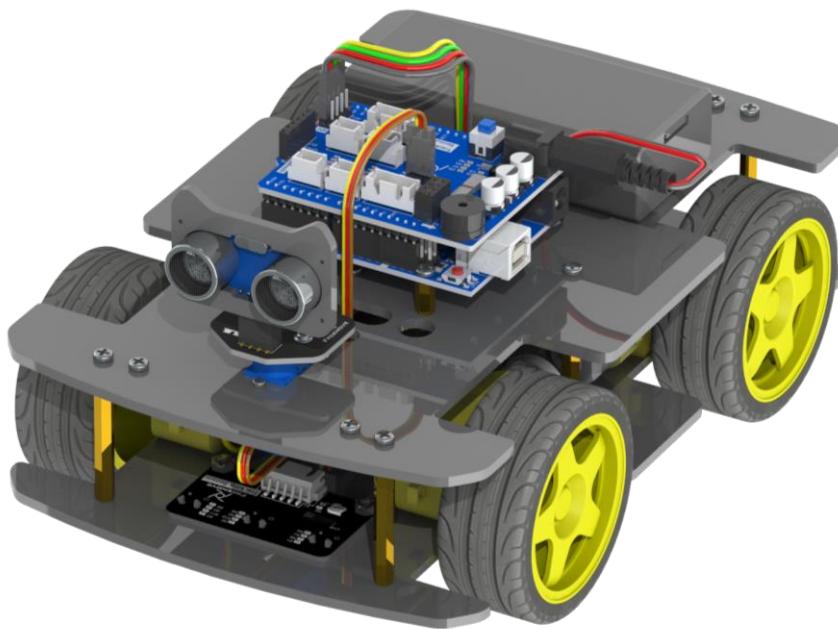
Installez le support M3 * 40 sur le panneau acrylique inférieur.



Installez le panneau acrylique supérieur à l'écart.

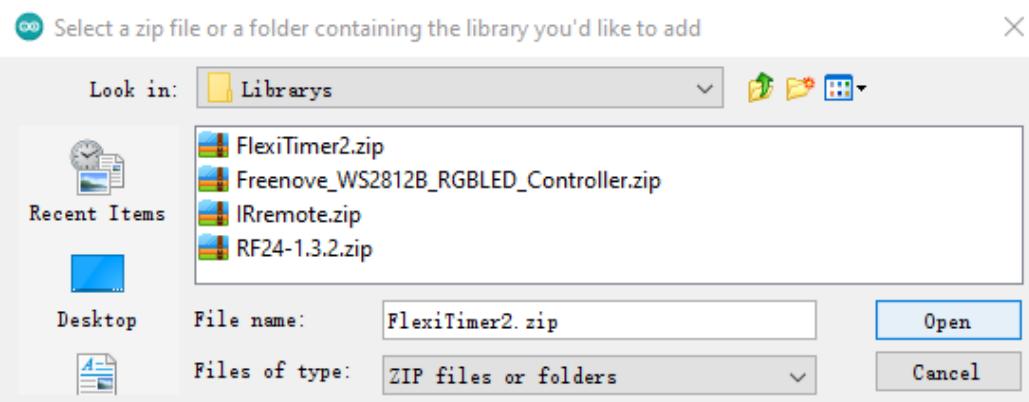
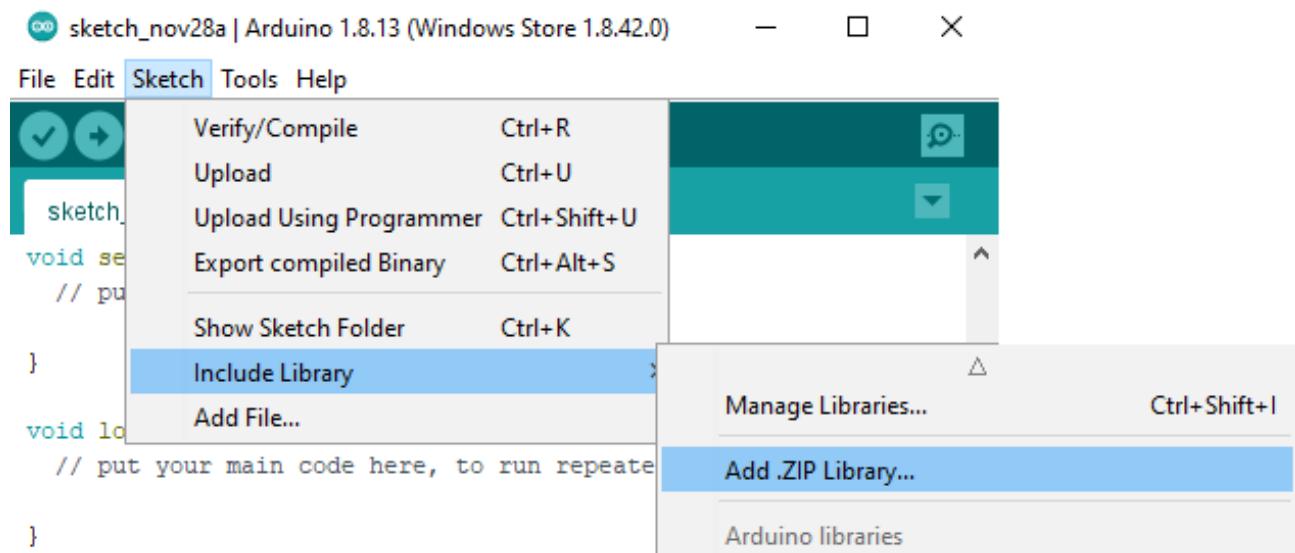


Maintenant, tout l'assemblage est terminé.



0.3 Comment jouer

Étape 1 Ajouter des bibliothèques

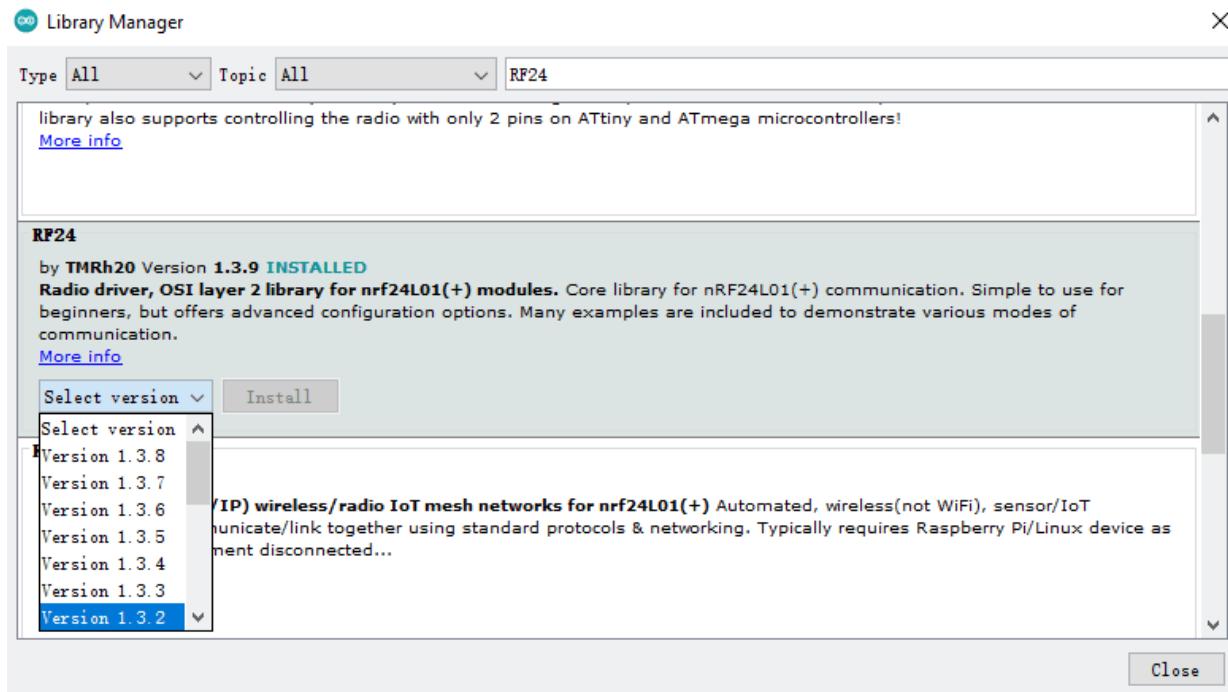


Ajouter **toutes les 4 bibliothèques** dans le dossier **Freenove_4WD_Car_Kit \ Librarys** un par un.

The screenshot shows a Windows File Explorer window. The path is displayed as: "Company (G:) > GitHub > Freenove_4WD_Car_Kit > Librarys". The "Librarys" folder is highlighted with a red box. Inside the folder, there are four files listed: FlexiTimer2.zip, Freenove_WS2812B_RGBLED_Controller.zip, IRremote.zip, and RF24-1.3.2.zip. The table below provides a detailed view of these files:

Name	Date modified	Type
FlexiTimer2.zip	11/27/2020 4:49 PM	360压缩 ZIP
Freenove_WS2812B_RGBLED_Controller.zip	7/11/2020 10:02 AM	360压缩 ZIP
IRremote.zip	7/11/2020 10:02 AM	360压缩 ZIP
RF24-1.3.2.zip	12/1/2020 2:58 PM	360压缩 ZIP

Si vous avez installé des bibliothèques RF24, vous devez sélectionner pour installer la version 1.3.2



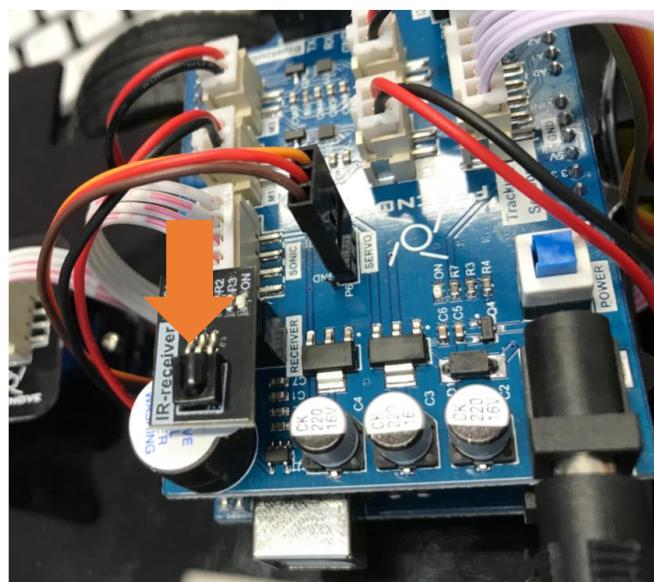
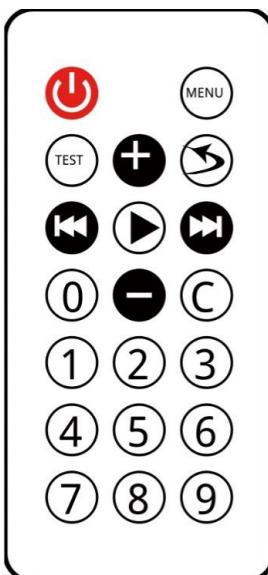
Étape 2

Si tu veux

Télécommande IR c

Installer IR re

télécharger différents codes.



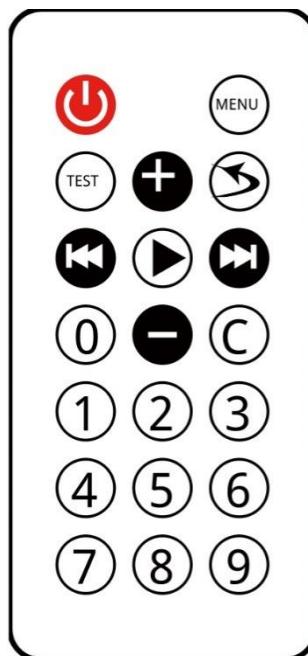
Téléchargez le code suivant sur le panneau de la voiture. **Veuillez supprimer BluetoothModule lorsque vous téléchargez le code.**

[Freenove_4WD_Car_Kit \ Sketches \ 04.4_One_Code_Multifunctional_IR_Remote_Car.ino](#)

Une fois le téléchargement effectué, vous pouvez utiliser la télécommande IR pour contrôler la voiture.

Une fois le code téléchargé, vous contrôlez la voiture et d'autres fonctions. La table c:

le contrôle de
e suivant



Graphique clé	Définition de la clé	Code clé	Fonction
	IR_REMOTE_KEYCODE_UP	0xFF02FD	avancer
	IR_REMOTE_KEYCODE_DOWN	0xFF9867	reculer
	IR_REMOTE_KEYCODE_LEFT	0xFFE01F	Tournez à gauche
	IR_REMOTE_KEYCODE_RIGHT	0xFF906F	Tournez à droite
	IR_REMOTE_KEYCODE_CENTER	0xFFA857	Activer le buzzer
	IR_REMOTE_KEYCODE_1	0xFF30CF	Faites fonctionner la LED en mode 1 pour faire défiler la couleur de l'arc-en-ciel.
	IR_REMOTE_KEYCODE_4	0xFF10EF	Faire fonctionner la LED en mode 2, en changeant la couleur de la LED d'eau
	IR_REMOTE_KEYCODE_2	0xFF18E7	La couleur de la barre LED change plus rapidement. La couleur provient de ColorWheel.
	IR_REMOTE_KEYCODE_3	0xFF7A85	La couleur de la barre LED change plus lentement.
	IR_REMOTE_KEYCODE_5	0xFF38C7	La période de cycle de la barre LED est diminuée et la barre de LED change à une vitesse plus rapide

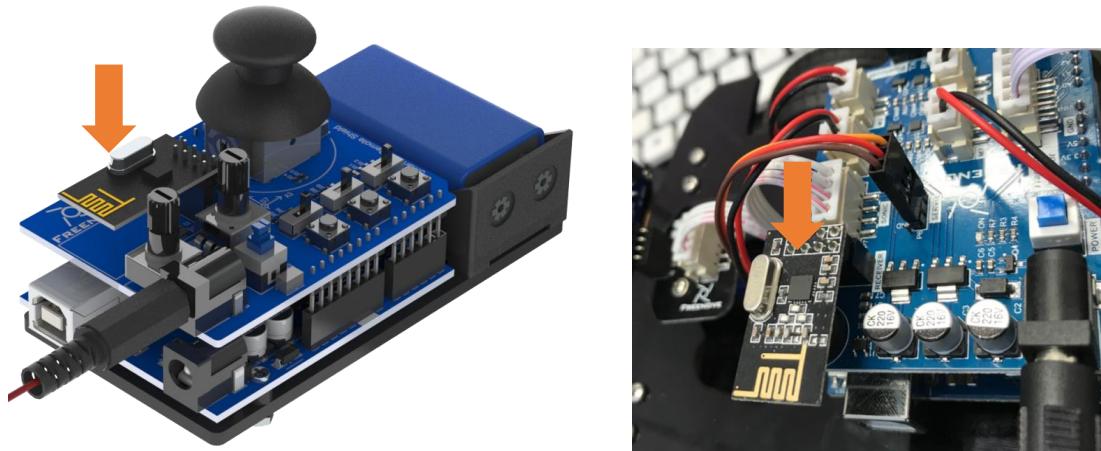
(6)	IR_REMOTE_KEYCODE_6	0xFF5AA5 La période de cycle de la barre LED est augmentée et la barre de LED change à une vitesse plus lente
-----	---------------------	---

Télécommande RF

Téléchargez le tutoriel et le code pour **Assembler** télécommande.

https://github.com/Freenove/Freenove_Remote_Control_Kit/raw/master/Tutorial.pdf

Retirez le revêtement IR et installez le module RF.



Vous devez supprimer le module de module Bluetooth lorsque vous téléchargez le code.

Téléchargez le code suivant sur le panneau de la voiture.

Freenove_4WD_Car_Kit \ Sketches \ 05.5_One_Code_Multifunctional_RF24_Remote_Car

Téléchargez le code suivant sur la télécommande RF.

Freenove_4WD_Car_Kit \ Sketches \ 05.1_RF24_Remote_Controller

Une fois le téléchargement effectué, vous pouvez utiliser la télécommande RF pour contrôler la voiture.

Changer de mode

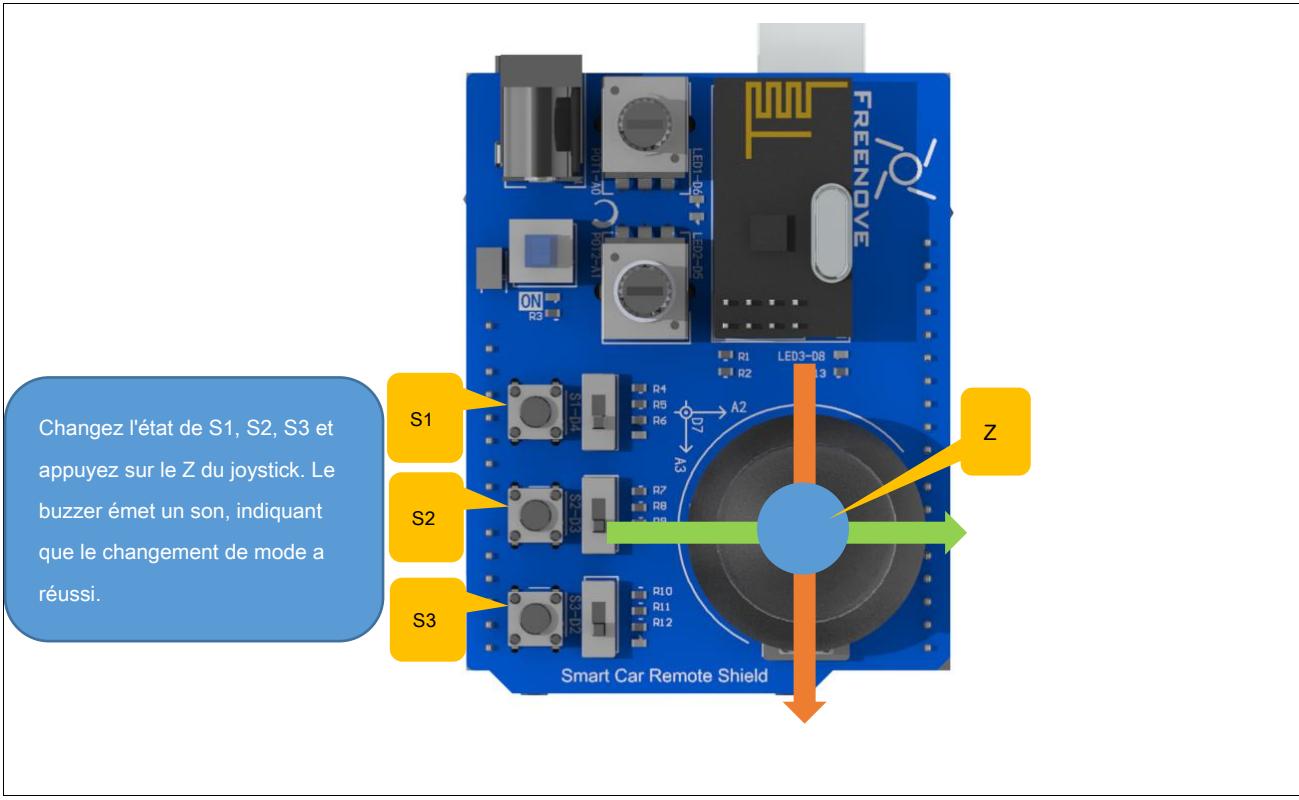
1, changez l'état du commutateur de S1, S2 et S3, et la voiture cessera de bouger.

2, appuyez sur l'axe Z du joystick et le buzzer B retentit pour indiquer que le mode est commuté avec succès.

Le tableau suivant montre les modes indiqués par les différents états des trois commutateurs S1, S2 et S3. La LED à côté de l'interrupteur s'allume pour indiquer l'état ON et OFF des interrupteurs. Les trois commutateurs peuvent former $2 \times 2 \times 2 = 8$ modes.

S1	S2	S3	Numéro de mode	Mode
ON	ON	ON	0	Aucun
ON	ON	OFF	1	Calibrer le mode servo
ON	OFF	ON	2	Aucun
ON	OFF	OFF	3	Mode évitement d'obstacles
OFF	ON	ON	4	Aucun
OFF	ON	OFF	5	Mode de suivi de ligne

OFF OFF ON 6		Changer de mode LED
OFF OFF OFF 7		Mode de contrôle manuel / Mode par défaut



Mode 0, 2, 4

Réserve. Nous ne leur avons pas attribué de fonctions.

Mode 1-Calibrer le servo

Si votre servo n'est pas monté avec précision à 90 degrés, vous pouvez utiliser ce mode pour un réglage fin (+ -10 degrés).

Dans ce mode, vous pouvez régler le potentiomètre 2 (POT2) pour affiner l'angle du servo. Lorsque vous ajustez le servo à l'angle correct, appuyez sur l'axe Z du joystick pour enregistrer les données d'étalonnage dans l'EEPROM. Il sera enregistré de manière permanente à moins qu'il ne soit modifié.

Mode 3-évitement obstical, Mode mode de suivi 5 lignes

Ces deux modes ont été appris séparément dans le projet précédent, et leur logique d'exécution et leurs codes sont cohérents avec le projet précédent.

La différence est que dans ce projet, la voiture peut répondre à tout moment aux commandes de la télécommande. Par conséquent, dans ce projet, il est toujours nécessaire de communiquer avec la télécommande dans ces deux modes. Lorsque le signal de la télécommande est déconnecté, la voiture s'arrête. Par conséquent, la communication normale entre la télécommande et la voiture doit être maintenue à tout moment. De mauvaises conditions de communication peuvent entraîner un fonctionnement anormal de ces deux modes.

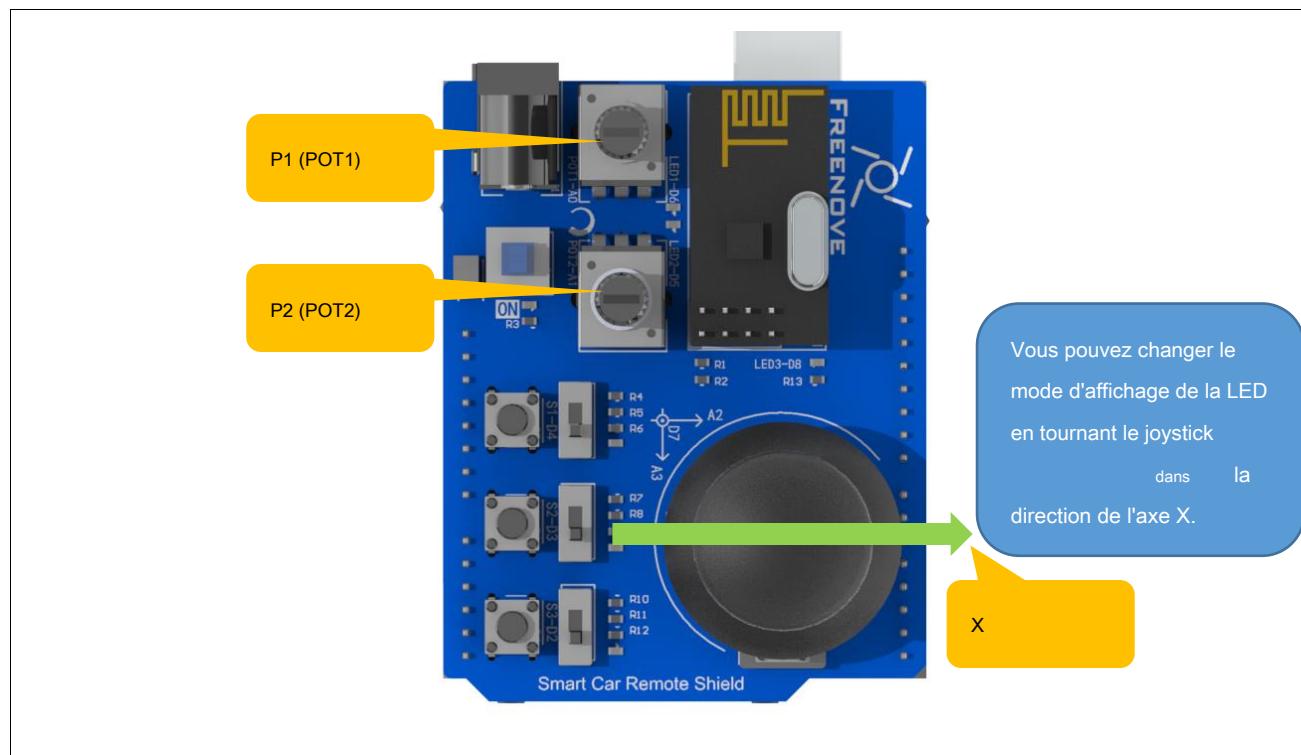
Mode d'affichage LED à 6 commutateurs

Il existe trois modes d'affichage pour les LED de la voiture, qui sont un arc-en-ciel à 0 écoulement, 1 LED à eau qui coule, 2 clignotements. Dans ce mode, le mode d'affichage de la LED peut être commuté.

Après être entré dans ce mode,

Déplacez le joystick le long de la direction positive de son axe X pour faire passer la LED au mode suivant. Déplacez le joystick dans le sens négatif de son axe X pour faire passer la LED au mode précédent.

Dans n'importe quel mode, les LED peuvent être réglées avec les potentiomètres P1 et P2. P1 est utilisé pour ajuster le changement de couleur de la LED, et P2 est utilisé pour ajuster la fréquence de changement de LED.



Mode 7 - mode manuel à distance

Ce mode est le mode manuel à distance et est le mode par défaut. Ce mode est cohérent avec le projet précédent "RF_Remote_Car". Utilisez le joystick pour contrôler pour avancer, reculer et tourner à gauche, tourner à droite.

Les chapitres suivants vous apprendront à contrôler les composants.

Chapitre 1 Contrôle des composants de base

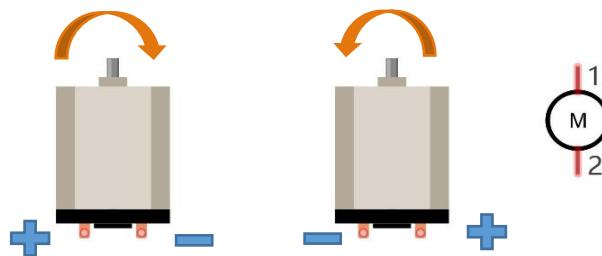
Ce chapitre présentera comment contrôler le moteur, le buzzer, la LED RVB et obtenir le niveau de la batterie.

Si vous avez des préoccupations, n'hésitez pas à nous contacter via support@freenove.com

1.1 Moteur

Lorsque le moteur est connecté à l'alimentation, il tourne dans un sens. Inversez la polarité de l'alimentation, le moteur tournera dans le sens opposé.

Et la vitesse du moteur dépend de la tension entre deux extrémités. Plus la tension est élevée, plus la vitesse est élevée.



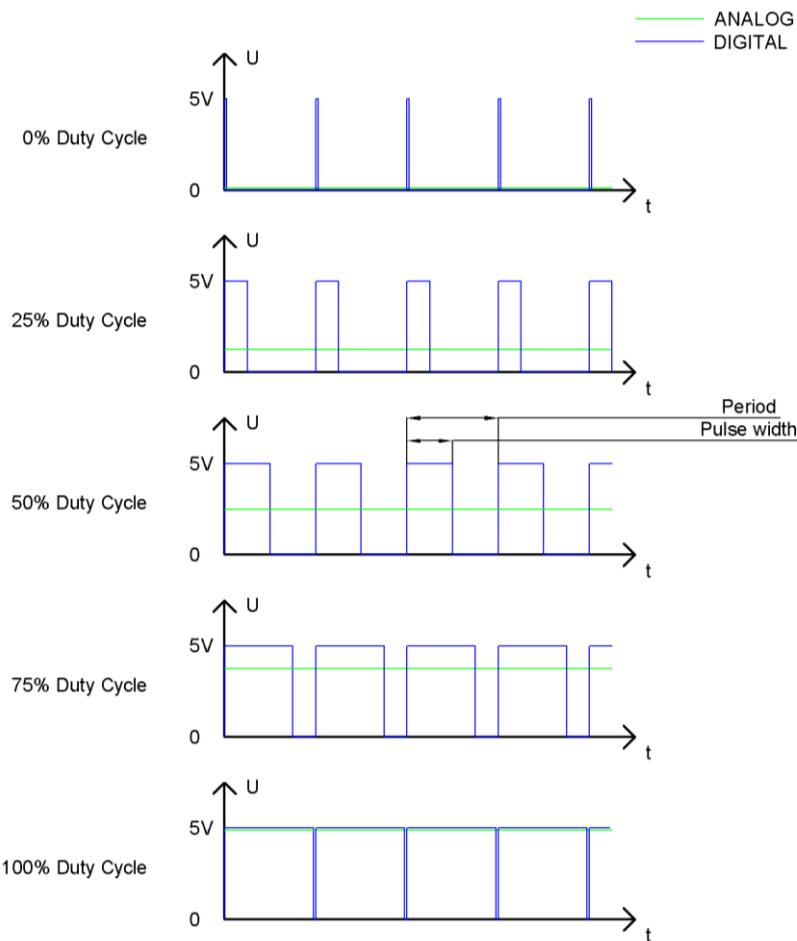
PWM

PWM, Pulse Width Modulation, utilise des broches numériques pour envoyer certaines fréquences d'ondes carrées, c'est-à-dire la sortie de niveaux élevés et de niveaux bas, qui durent alternativement pendant un certain temps. Le temps total pour chaque ensemble de niveaux élevés et de niveaux bas est généralement fixe, ce qui est appelé la période (l'inverse de la période est la fréquence). Le temps des sorties de haut niveau est généralement appelé «largeur d'impulsion», et le rapport cyclique est le pourcentage du rapport de la durée d'impulsion, ou largeur d'impulsion (PW) à la période totale (T) de la forme d'onde.

Plus la sortie des niveaux élevés dure longtemps, plus le rapport cyclique est grand et plus la tension correspondante dans le signal analogique sera élevée.

Les figures suivantes montrent comment la tension du signal analogique varie entre 0V-5V

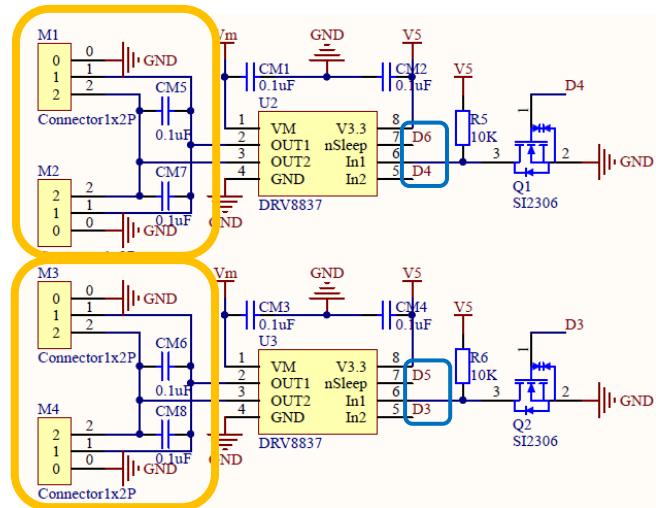
(le niveau haut est 5V) correspondant à la largeur d'impulsion 0% -100%:



Plus le cycle de service PWM est long, plus la puissance de sortie sera élevée. Maintenant que nous comprenons cette relation, nous pouvons utiliser PWM pour contrôler la luminosité d'une LED ou la vitesse du moteur à courant continu, etc.

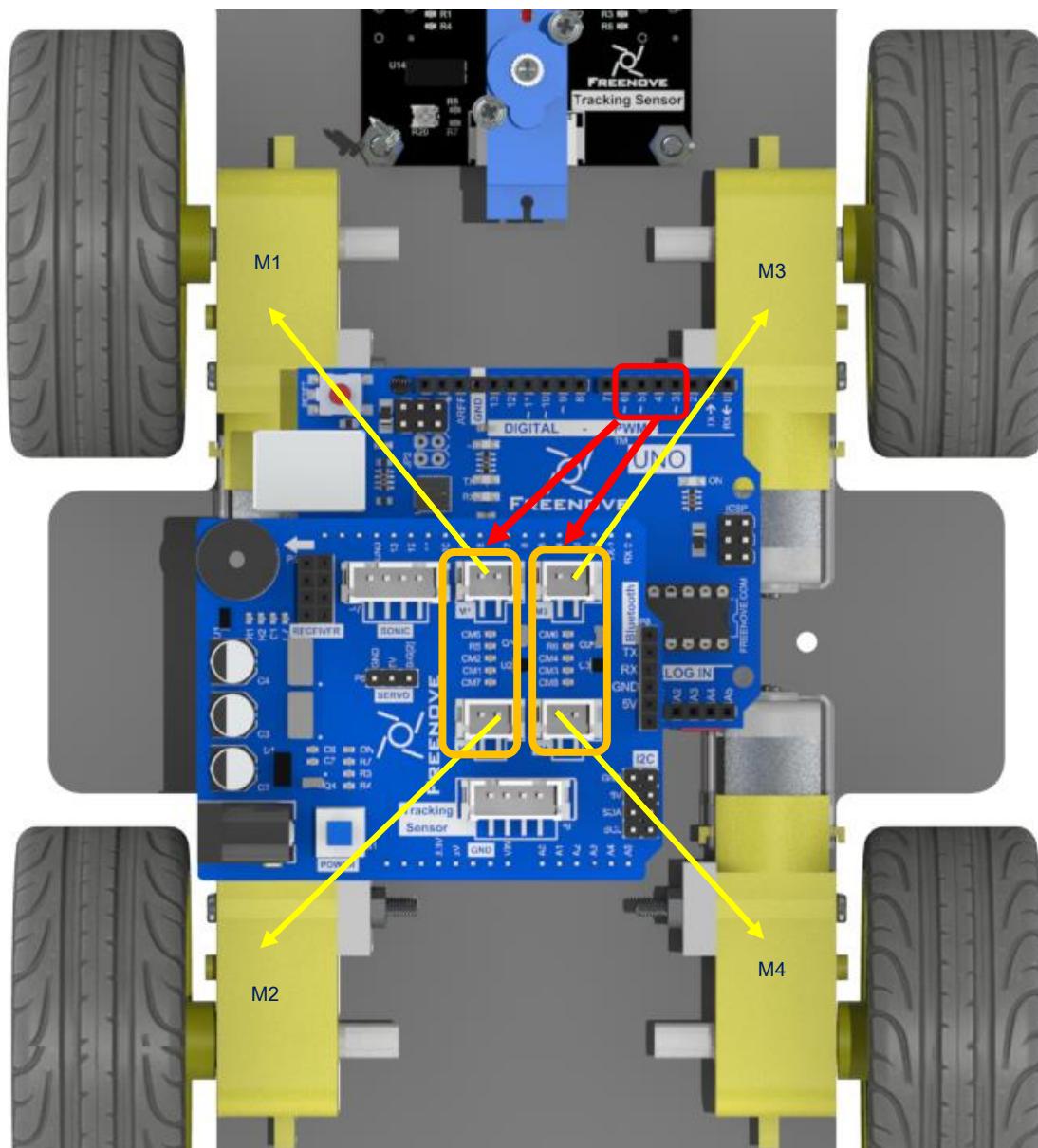


Dans cette voiture, M1 et M2 sont connectés en parallèle; M3 et M4 sont connectés en parallèle. Le schéma est ci-dessous:



Motor [M1, M2, M3, M4]

Les broches D3 et D5 de la carte de commande contrôlent respectivement M3 et M4. Les broches D4 et D6 de la carte contrôlent M1 et M2.



Code

01.1.1_RunMotor_Left_Wheel

Si vous ne savez pas comment télécharger du code sur Arduino, vous pouvez vous référer à [Première utilisation](#).

Vous devez supprimer le module de module Bluetooth lorsque vous téléchargez le code.

Ensuite, coupez l'alimentation. Et connectez la carte de contrôle à l'ordinateur avec un câble USB. Téléchargez le code dans le dossier: Sketches \ 01.1.1_RunMotor_Left_Wheel

Ensuite, déconnectez le câble USB et placez votre voiture dans un endroit avec suffisamment d'espace pour se déplacer. Ou faites lever la tête de la voiture et observez la roue. **Le code suivant nécessite également cette étape. Allumez la voiture.** Ensuite, les moteurs et les roues gauches de la voiture tourneront d'avant en arrière. Et tu peux

voir les vitesses sont différentes. Vous pouvez éteindre la voiture pour arrêter cela.

De la section précédente, nous avons su que M1 et M2 sont connectés en parallèle et que M3 et M4 sont connectés en parallèle. Cela signifie que **vous ne pouvez contrôler que les moteurs du même côté ensemble mais** pas séparément. Habituellement, les moteurs du même côté n'ont pas besoin de tourner dans des directions différentes.

Le code est ci-dessous:

```
1      # définir PIN_DIRECTION_RIGHT
2      # définir PIN_DIRECTION_LEFT 4
3      # définir PIN_MOTOR_PWM_RIGHT 5
4      # définir PIN_MOTOR_PWM_LEFT 6
5
6      néant installer() {
sept    pinMode ( PIN_DIRECTION_LEFT , PRODUCTION);
8      pinMode ( PIN_MOTOR_PWM_LEFT , PRODUCTION);
9      pinMode ( PIN_DIRECTION_RIGHT , PRODUCTION);
dix     pinMode ( PIN_MOTOR_PWM_RIGHT , PRODUCTION);
11
12
13      néant boucle() {
14          // Les moteurs de gauche tournent dans un sens
15          digitalWrite ( PIN_DIRECTION_LEFT , HAUTE);
16          analogWrite ( PIN_MOTOR_PWM_LEFT , 100);
17          retard (1000);           // délai (ms), 1000ms = 1s
18
19          analogWrite ( PIN_MOTOR_PWM_LEFT , 0); //Arrêtez
20          retard (1000);
21
22          // le moteur gauche tourne dans le sens opposé
23          digitalWrite ( PIN_DIRECTION_LEFT , FAIBLE);
24          analogWrite ( PIN_MOTOR_PWM_LEFT , 255);
25          retard (1000);
26
27          analogWrite ( PIN_MOTOR_PWM_LEFT , 0); //Arrêtez
28          retard (1000);
29      }
```

Dans le code, le code suivant est utilisé pour définir les broches.

```
1 # définir PIN_DIRECTION_RIGHT 3  
2 # définir PIN_DIRECTION_LEFT 4  
3 # définir PIN_MOTOR_PWM_RIGHT 5  
4 # définir PIN_MOTOR_PWM_LEFT 6
```

Comme vous pouvez le voir dans le schéma ci-dessus, la broche des moteurs ne peut pas être modifiée.

définir AB, cela signifie que A est B. Utilisez A au lieu de B pour un entretien facile et une lecture facile.

Vous pouvez également définir des broches de type int, comme int PIN_DIRECTION_RIGHT = 3; Mais cela nécessite plus d'espace RAM.

Habituellement, il existe deux fonctions principales de base pour le code Arduino, void setup () et void loop (). void setup(){

}

La fonction setup () est appelée au démarrage d'une esquisse, utilisée pour initialiser les variables, les modes de broche, commencer à utiliser les bibliothèques, etc.

La fonction setup () ne courir qu'une seule fois, après chaque mise sous tension ou réinitialiser de la carte Arduino. boucle void () {}

Cette fonction va boucler consécutivement. Le code de cette fonction sera exécuté encore et encore...

Il existe d'autres fonctions intégrées par Arduino. pinMode (broche,

mode)

Configure la broche spécifiée pour qu'elle se comporte comme une entrée ou une sortie. Paramètres

pin: numéro de broche Arduino pour définir le mode. mode:

INPUT, OUTPUT ou INPUT_PULLUP. digitalWrite (broche,

valeur)

Écrivez une valeur HIGH ou LOW sur une broche numérique.

Si la broche a été configurée comme OUTPUT avec pinMode (), sa tension sera réglée à la valeur correspondante: 5V (ou 3,3V sur les cartes 3,3V) pour HIGH, 0V (masse) pour LOW.

Paramètres

pin: numéro de broche Arduino.

valeur: HIGH ou LOW. analogWrite

(broche, valeur)

Écrit une valeur analogique (onde PWM) sur une broche.

Vous n'avez pas besoin d'appeler pinMode () pour définir la broche comme sortie avant d'appeler analogWrite (). Paramètres

pin: broche Arduino sur laquelle écrire. Types de données autorisés: int.

valeur: le rapport cyclique: entre 0 (toujours désactivé) et 255 (toujours activé). Types de données autorisés: int. Pour plus de

détails, veuillez consulter: <https://www.arduino.cc/reference/>

Notez que pour analogWrite (broche, valeur) des moteurs pour cette voiture, la valeur maximale est de 255. Comme le moteur nécessite une certaine tension pour fonctionner, et analogwrite (broche, 1) ne répond pas à cette tension requise., Selon tension différente, la valeur qui fait fonctionner la voiture sera beaucoup plus grande que analogwrite (broche, 1).

01.1.2_RunMotor_Right_Wheel

Vous devez supprimer le module de module Bluetooth lorsque vous téléchargez le code.

Téléchargez le code dans le dossier: Sketches \ 01.1.2_RunMotor_Right_Wheel.

Ensuite, placez votre voiture dans un endroit avec suffisamment d'espace pour se déplacer. Mettez la voiture sous tension. Ensuite, les bons moteurs et roues de la voiture tourneront d'avant en arrière. Et vous pouvez voir que les vitesses sont différentes.

Parce que les moteurs gauche et droit sont en miroir. Lorsque les moteurs tournent dans le même sens, la voiture réagit en sens inverse.

Le code est ci-dessous:

```

1 # définir PIN_DIRECTION_RIGHT 3
2 # définir PIN_DIRECTION_LEFT 4
3 # définir PIN_MOTOR_PWM_RIGHT 5
4 # définir PIN_MOTOR_PWM_LEFT 6
5
6 néant installer() {
sept pinMode ( PIN_DIRECTION_LEFT , PRODUCTION);
8 pinMode ( PIN_MOTOR_PWM_LEFT , PRODUCTION);
9 pinMode ( PIN_DIRECTION_RIGHT , PRODUCTION);
dix pinMode ( PIN_MOTOR_PWM_RIGHT , PRODUCTION);
11 }
12
13 néant boucle() {
14 // Les moteurs droits tournent dans un sens
15 digitalWrite ( PIN_DIRECTION_RIGHT , HAUTE);
16 analogWrite ( PIN_MOTOR_PWM_RIGHT , 100);
17 retard (1000);
18 analogWrite ( PIN_MOTOR_PWM_RIGHT , 0);
19 retard (1000);
20
21 // Les moteurs droits tournent dans le sens opposé
22 digitalWrite ( PIN_DIRECTION_RIGHT , FAIBLE);
23 analogWrite ( PIN_MOTOR_PWM_RIGHT , 255);
24 retard (1000);
25 analogWrite ( PIN_MOTOR_PWM_RIGHT , 0);
26 retard (1000);
27 }
28

```

01.1.3_Car_Move_and_Turn

Vous devez supprimer le module de module Bluetooth lorsque vous téléchargez le code.

Téléchargez le code dans le dossier: Sketches \ 01.1.3_Car_Move_and_Turn.

Ensuite, placez votre voiture dans un endroit avec suffisamment d'espace pour se déplacer. Mettez la voiture sous tension. Ensuite, la voiture va avancer et reculer, puis tourner à gauche et tourner à droite. Après s'être arrêté pendant 2 secondes, il répète les actions. Le code est ci-dessous:

```

1 # définir PIN_DIRECTION_RIGHT 3
2 # définir PIN_DIRECTION_LEFT 4
3 # définir PIN_MOTOR_PWM_RIGHT 5
4 # définir PIN_MOTOR_PWM_LEFT 6
5
6 néant installer() {
sept pinMode ( PIN_DIRECTION_LEFT , PRODUCTION);
8 pinMode ( PIN_MOTOR_PWM_LEFT , PRODUCTION);
9 pinMode ( PIN_DIRECTION_RIGHT , PRODUCTION);

```

```
dix
11 }
12
13 néant boucle() {
14 //Avance
15 motorRun (200, 200);
16 retard (1000);
17
18 //Reculer
19 motorRun (-200, -200);
20 retard (1000);
21
22 //Tournez à gauche
23 motorRun (-200, 200);
24 retard (1000);
25
26 //Tournez à droite
27 motorRun (200, -200);
28 retard (1000);
29
30 //Arrêtez
31 motorRun (0, 0);
32 retard (2000);
33 }
34
35 néant motorRun ( int speedl , int speedr ) {
36     int dirL = 0, dirR = 0;
37     si ( speedl > 0 ) {dirL = 0;
38
39     }
40     autre {
41         dirL = 1;
42         speedl = - speedl ;
43     }
44     si ( speedr > 0 ) {dirR = 1;
45
46     }
47     autre {
48         dirR = 0;
49         speedr = - speedr ;
50     }
51     digitalWrite ( PIN_DIRECTION_LEFT , dirL);
52     digitalWrite ( PIN_DIRECTION_RIGHT , dirR);
53     analogWrite ( PIN_MOTOR_PWM_LEFT , speedl );
```

```

54   analogWrite ( PIN_MOTOR_PWM_RIGHT , speedr );
55 }
56

```

Maintenant nous utiliserons `motorRun (200, -200)` comme exemple pour introduire cette fonction.

```

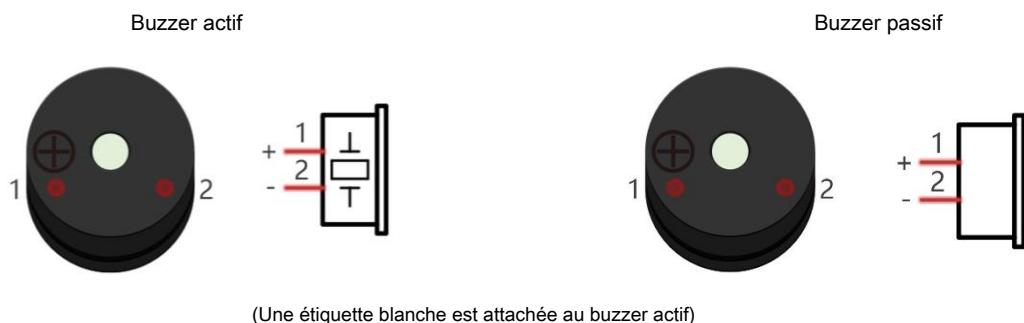
35   néant motorRun ( int speedl , int speedr ) {           // speedl = 200, speedr = -200
36       int dirL = 0, dirR = 0;
37       si ( speedl > 0) {           // speedl = 200> 0, ou
38           dirL = 0;             // donc dirL = 0
39       }
40       autre {
41           dirL = 1;
42           speedl = - speedl ;
43       }
44       si ( speedr > 0) {dirR = 1;           // speedr = -200 <0
45
46   }
47       autre {
48           dirR = 0;             // so dirR = 0
49           speedr = - speedr ;           // puis speedr = 200
50       }
51       digitalWrite ( PIN_DIRECTION_LEFT , dirL);           // digitalWrite (PIN_DIRECTION_LEFT, 0);
52       digitalWrite ( PIN_DIRECTION_RIGHT , dirR); // digitalWrite (PIN_DIRECTION_LEFT, 0);
53       analogWrite ( PIN_MOTOR_PWM_LEFT , speedl ); // analogWrite (PIN_MOTOR_PWM_LEFT, 200);
54       analogWrite ( PIN_MOTOR_PWM_RIGHT , speedr ); // analogWrite (PIN_MOTOR_PWM_RIGHT, 200)
55   }

```

Les moteurs gauche et droit sont **en miroir**. Lorsque les moteurs des deux côtés tournent dans le même sens, la voiture tourne. Afin de faire avancer ou reculer une voiture, nous devons régler le moteur sur deux côtés avec une direction différente. Donc `motorRun (200, -200)` fera tourner la voiture à droite.

1.2 Buzzer et niveau de batterie

Un buzzer est un composant audio. Il existe des types de buzzers actifs et passifs. Les buzzers actifs ont un oscillateur à l'intérieur, ceux-ci retentiront tant que l'alimentation est fournie. Les buzzers passifs nécessitent un signal d'oscillateur externe (utilisant généralement PWM avec des fréquences différentes) pour produire un son.

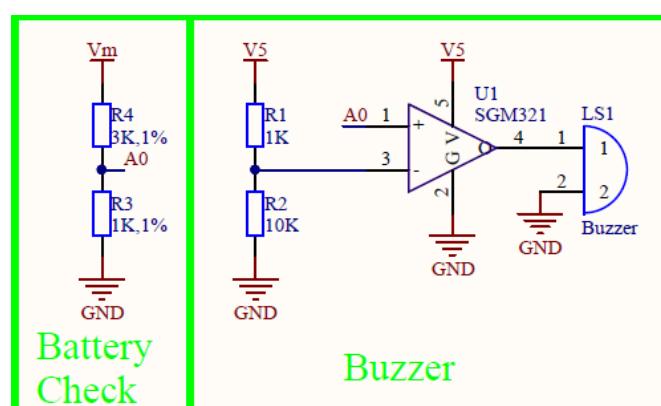


Les buzzers actifs sont plus faciles à utiliser. Généralement, ils ne peuvent produire qu'une fréquence sonore spécifique.

Les buzzers passifs nécessitent un circuit externe pour produire des sons, mais ils peuvent être contrôlés pour produire des sons de différentes fréquences. La fréquence de résonance du buzzer passif est de 2 kHz, ce qui signifie que le buzzer passif est le plus fort lorsque la fréquence de résonance est de 2 kHz.

Dans cette voiture, le buzzer est un buzzer actif. Donc, il ne peut produire qu'une fréquence sonore spécifique. La tonalité de fonction () d'Arduino ne peut pas être utilisée pour ce type de buzzer.

Puisque la carte de contrôle a un nombre limité d'E / S. Le buzzer et le niveau de la batterie ne sont pas utilisés fréquemment. Nous utilisons donc une E / S de la carte de contrôle contrôlant le buzzer et détectant le niveau de la batterie à différents moments. Le schéma du buzzer et de la batterie est ci-dessous:



Comme vous pouvez le voir, ils utilisent le même IO A0.

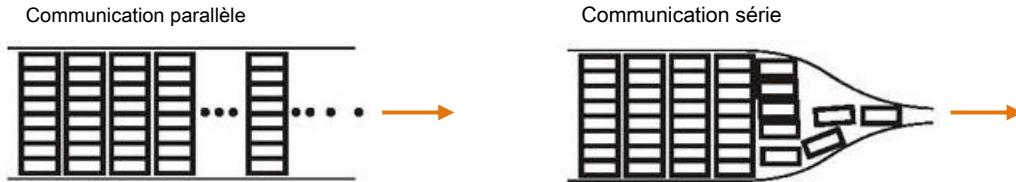
Le niveau de la batterie doit être $V_m = 4 * V_0$. V_0 est la tension détectée par A0 de la carte de commande. Le niveau maximum de la batterie est de 8,4 V ($V_0 = 2,1$ V).

Le bon circuit signifie que si $V_0 > 4,5$ V, le buzzer émettra un son.

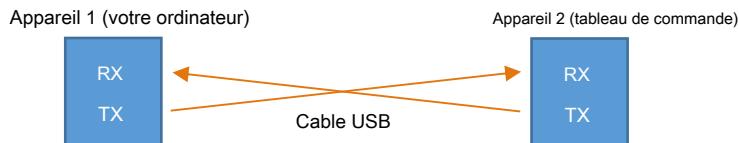
Si V_0 est compris entre (0 ~ 2,1 v), il est utilisé pour la batterie. Si $V_0 > 4,5$ V, il est utilisé pour le buzzer.

Communication série

La communication série utilise un câble de données pour transférer les données un bit par un autre à tour de rôle. La communication parallèle signifie que les données sont transmises simultanément sur plusieurs câbles. La communication série ne prend que quelques câbles pour échanger des informations entre les systèmes, ce qui est particulièrement adapté pour les ordinateurs aux ordinateurs, la communication longue distance entre les ordinateurs et les périphériques. La communication parallèle est plus rapide, mais elle nécessite plus de câbles et un coût plus élevé, elle n'est donc pas appropriée pour les communications longue distance.



La communication série fait généralement référence au récepteur / émetteur asynchrone universel (UART), qui est couramment utilisé dans la communication de circuits électroniques. Il dispose de deux lignes de communication, l'une est responsable de l'envoi des données (ligne TX) et l'autre de la réception des données (ligne RX). Les connexions de communication série de deux appareils sont les suivantes:



Pour la communication série, le **la vitesse de transmission des deux côtés doit être la même**. Les débits en bauds couramment utilisés sont 9600 et 115200.

L'ordinateur identifie les périphériques série connectés à votre ordinateur comme COMx. Nous pouvons utiliser la fenêtre Serial Monitor du logiciel Arduino pour communiquer avec la carte de contrôle Freenove.

Code

01.2.1_Buzzer

Vous devez supprimer le module de module Bluetooth lorsque vous téléchargez le code.

Téléchargez le code dans Sketches \ 01.2.1_Buzzer. Puis tournez l'interrupteur de la voiture. Vous entendrez b-bbbb ---- bbbb ---- bbbb ----

Le code dans setup () n'est exécuté qu'une seule fois. Le code en boucle () sera exécuté de manière circulaire.

```

1 # définir PIN_BATTERY      A0
2 # définir PIN_BUZZER       A0
3
4 néant installer() {
5   pinMode ( PIN_BUZZER , PRODUCTION);
6   digitalWrite ( PIN_BUZZER , HAUTE);
7   retard (100);
8   digitalWrite ( PIN_BUZZER , FAIBLE);
9 }
dix
11 néant boucle() {

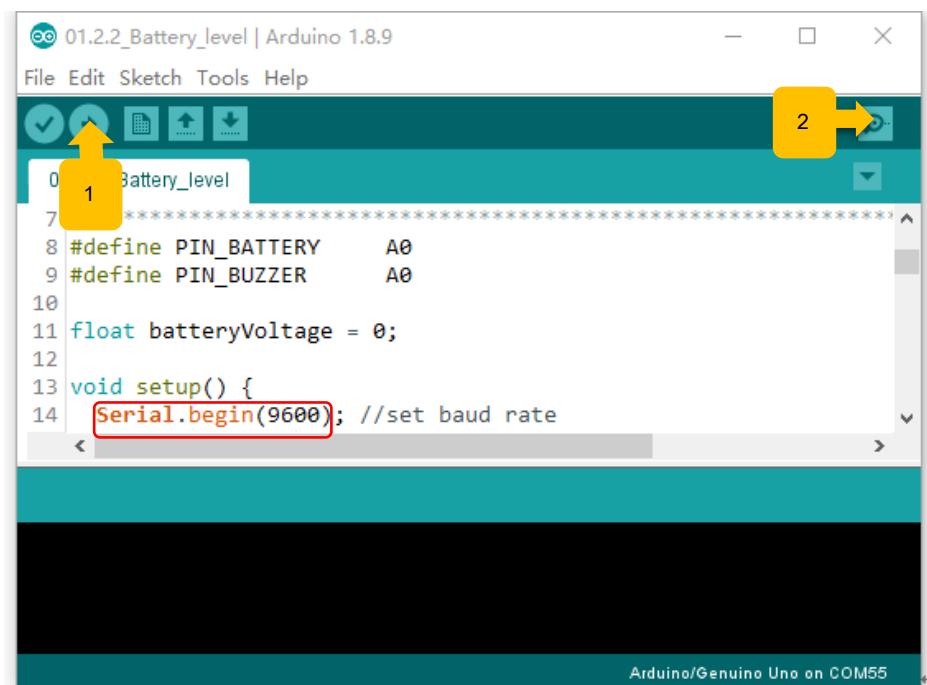
```

```
12 pour ( int i = 0; i <4; i ++) {  
13     digitalWrite ( PIN_BUZZER , HIGH); // active le délai du buzzer (100);  
14  
15     digitalWrite ( PIN_BUZZER , FAIBLE); // désactiver le délai du buzzer (100);  
16  
17 }  
18 retard (500);  
19 }
```

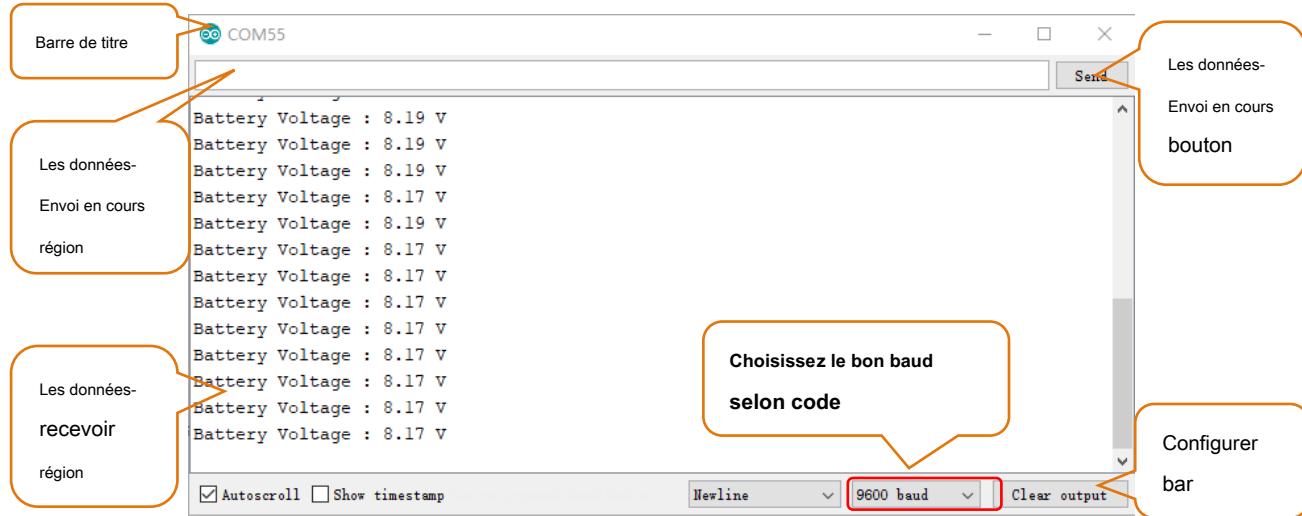
01.2.2_Battery_level

Téléchargez le code dans Sketches \ 01.2.2_Battery_level. Ne déconnectez pas le câble USB. Puis allumez l'interrupteur de la voiture.

Cliquez ensuite sur l'icône Serial Monitor pour ouvrir la fenêtre Serial Monitor.



Ensuite, vous entendrez le buzzer émettre deux sons. Et puis voyez la fenêtre Interface of Serial Monitor comme suit. Si vous ne pouvez pas l'ouvrir, assurez-vous que la carte de contrôle Freenove a été connectée à l'ordinateur, et choisissez le bon port série dans la barre de menu "Outils-Port".



Le code est ci-dessous:

```

1 # définir PIN_BATTERY          A0
2 # définir PIN_BUZZER           A0
3
4 flotte batteryVoltage = 0;
5
6 néant installer() {
7     Serial.begin (9600); // définir le débit en bauds
8     pinMode ( PIN_BUZZER , PRODUCTION);
9
10    digitalWrite ( PIN_BUZZER , HAUTE);
11    retard (500);
12    digitalWrite ( PIN_BUZZER , FAIBLE);
13    retard (500);
14    digitalWrite ( PIN_BUZZER , HAUTE);
15    retard (500);
16    digitalWrite ( PIN_BUZZER , FAIBLE);
17 }
18
19 néant boucle() {
20     // Etant donné que le buzzer et la détection de la tension de la batterie utilisent la broche A0 ensemble, // le buzzer
21     // doit être désactivé lors de la lecture de la tension de la batterie, // sinon la tension de la batterie de lecture est
22     // incorrecte.
23
24     // Désactiver le buzzer
25     pinMode ( PIN_BUZZER , faux );
26     digitalWrite ( PIN_BUZZER , faux );
27
28     // impression de la tension de la batterie
29     si (getBatteryVoltage ()) {
30         Serial.print ( "Voltage de batterie : " );

```

```

31     Serial.print (batteryVoltage);
32     Serial.println ( "V" );
33 }
34 autre {
35     Serial.print ( "Port occupé! Veuillez désactiver le buzzer avant de lire la tension de la batterie." );
36 }
37 retard (1000);
38 }
39
40 booléen getBatteryVoltage () {
41     pinMode ( PIN_BATTERY , CONTRIBUTION );
42     int batteryADC = analogRead ( PIN_BATTERY );
43     si ( batterieADC <614) // 614/1023 * 5 = 3 V, tension raisonnable: <2,1 V
44     {
45         batteryVoltage = batteryADC / 1023.0 * 5.0 * 4;
46         retourne vrai ;
47     } autre {
48         revenir faux ;
49     }
50 }
```

Dans ce code, différents types de données sont mentionnés: float, bool, void. Si vous n'êtes pas familiarisé avec les types de données, reportez-vous à <https://www.arduino.cc/reference/>

VARIABLES

Arduino data types and constants.

Constants	Conversion	float	Variable Scope & Qualifiers
Floating Point Constants	(unsigned int)	int	const
Integer Constants	(unsigned long)	long	scope
HIGH LOW	byte()	short	static
INPUT OUTPUT INPUT_PULLUP	char()	size_t	volatile
LED_BUILTIN	float()	string	
true false	int()	unsigned char	Utilities
	long()	unsigned int	PROGMEM
	word()	unsigned long	sizeof()
		void	
		word	
	Data Types		
	String()		
	array		
	bool		
	boolean		
	byte		
	char		
	double		

La fonction est définie sous la forme: type de données nom de la fonction () .

S'il s'agit de void function name (), après son exécution, il ne retournera rien.

```

28 // impression de la tension de la batterie
29 si ( getBatteryVoltage () {
30     Serial.print ( "Voltage de batterie : " );
31     Serial.print ( batteryVoltage );
32     Serial.println ( "V" );
33 }
34 autre {
35     Serial.print ( "Port occupé! Veuillez désactiver le buzzer avant de lire la tension de la batterie." );
36 }
37 retard (1000);
38 }
39
40 boolengetBatteryVoltage () {
41     pinMode ( PIN_BATTERY , CONTRIBUTION );
42     int batteryADC = analogRead ( PIN_BATTERY );
43     si ( batterieADC <614 )                                // 614/1023 * 5 = 3 V, tension raisonnable: <2,1 V
44     {
45         batteryVoltage = batteryADC / 1023.0 * 5.0 * 4;
46         revenir vrai ;
47     } autre {
48         revenir faux ;
49     }
50 }
```

Le code est toujours exécuté ligne par ligne. Quand if (getBatteryVoltage ()) est exécuté, le programme se tournera pour exécuter bool getBatteryVoltage () {}. Après le retour, le programme exécutera le code dans if { }.

Serial.print ()

La syntaxe est ci-dessous:

Serial.print (val)

Serial.print (val, format)

Série: objet de port série. Consultez la liste des ports série disponibles pour chaque carte sur la page principale Série. val: la valeur à imprimer. Types de données autorisés: tout type de données.

La seule différence est que Serial.println () enverra un «\n». le contenu du prochain print () ou println () sera affiché dans une nouvelle ligne.

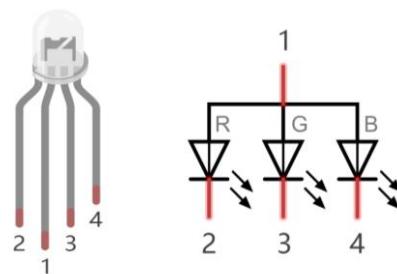
analogRead (broche)

Il mappera les tensions d'entrée entre 0 et la tension de fonctionnement (5V ou 3,3V) en valeurs entières entre 0 et 1023. Pour 5v par exemple, ADCvalue = analogRead (pin)

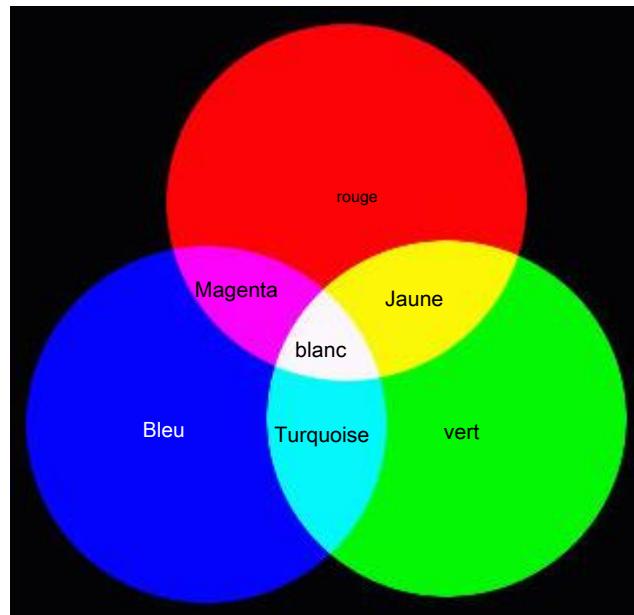
Si vous souhaitez obtenir la valeur de tension. Il doit être V = valeur ADC / 1023 * 5V.

1,3 RVB

Une LED RVB a 3 LED intégrées dans un composant LED. Il peut respectivement émettre de la lumière rouge, verte et bleue. Pour ce faire, il nécessite 4 broches (c'est également ainsi que vous l'identifiez). La broche longue (1) est la commune qui est la cathode (+) ou le fil positif, les 3 autres sont les anodes (-) ou les fils négatifs. Un rendu d'une LED RVB et de son symbole électronique est présenté ci-dessous. Nous pouvons faire en sorte que la LED RVB émette différentes couleurs de lumière et de luminosité en contrôlant les 3 anodes (2, 3 et 4) de la LED RVB



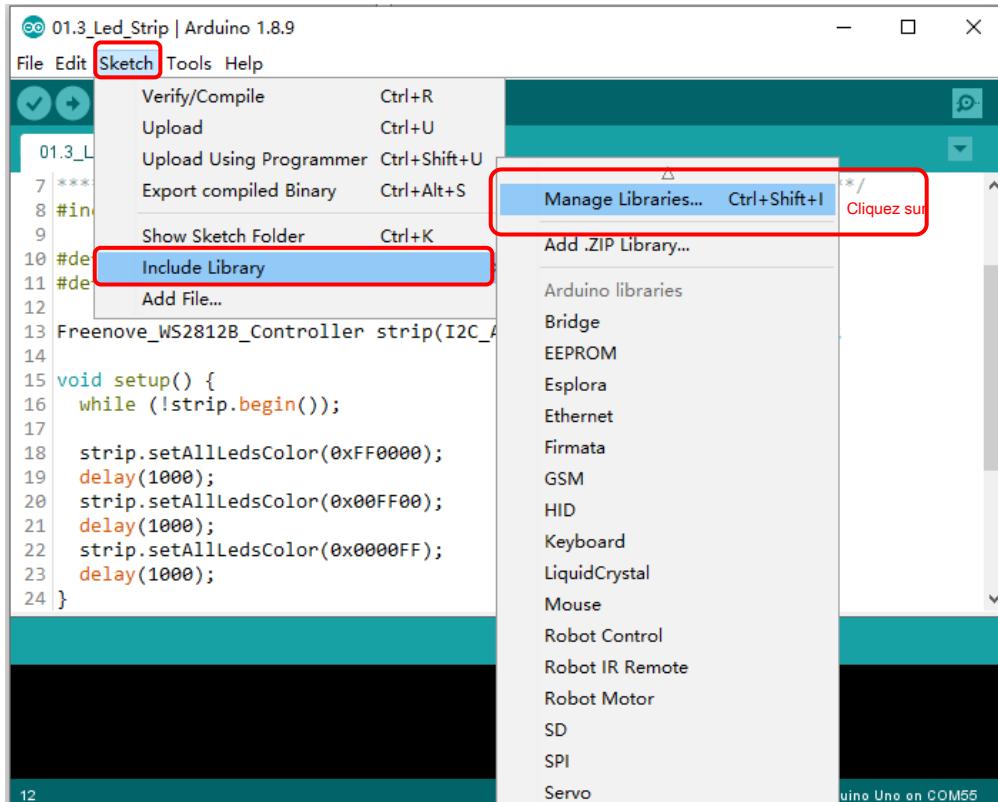
La lumière rouge, verte et bleue est appelée 3 couleurs primaires lorsque l'on parle de lumière (Remarque: pour les pigments tels que les peintures, les 3 couleurs primaires sont le rouge, le bleu et le jaune). Lorsque vous combinez ces trois couleurs primaires de lumière avec une luminosité variée, elles peuvent produire presque toutes les couleurs de lumière visible. Les écrans d'ordinateur, les pixels simples des écrans de téléphones portables, les lampes au néon, etc. peuvent tous produire des millions de couleurs en raison de ce phénomène.



Nous savons de la section précédente que, la carte de contrôle contrôle les LED pour émettre un total de 256 (0-255) luminosité différente avec PWM.

Ainsi, grâce à la combinaison de trois couleurs différentes de LED, la LED RVB peut émettre $256^3 = 16777216$ couleurs, 16 millions de couleurs.

Inclure la bibliothèque



Entrez "Freenove_WS2812B_RGBLED_Controller" et appuyez sur la touche "Entrée" pour rechercher. Trouve et **installer** la dernière version (maintenant c'est la version 1.0.0).



Ensuite, revenez en arrière et téléchargez le code.

Code

01.3_Led_Strip

Vous devez supprimer le module de module Bluetooth lorsque vous téléchargez le code.

Téléchargez le code dans Sketches \ 01.3_Led_Strip. Puis allumez la voiture.

Vous pouvez trouver une vue appropriée pour observer les couleurs des LED sur le panneau acrylique inférieur.



Vous verrez que toutes les LED affichent tour à tour le rouge, le vert, le bleu et le jaune. Ensuite, toutes les LED s'éteignent.

Ensuite, les 5 premières LED montrent différentes couleurs une par une. Ensuite, toutes les LED affichent la même couleur et la couleur change en douceur. Ensuite, différentes LED affichent des couleurs différentes et changent en douceur.

Le code est ci-dessous:

```
1 # comprendre "Freenove_WS2812B_RGBLED_Controller.h"
2
3 # définir I2C_ADDRESS 0x20
4 # définir LEDS_COUNT      dix // il définit le nombre de LED
5
6 Bande Freenove_WS2812B_Controller ( I2C_ADDRESS , LEDS_COUNT , TYPE_GRB); // initialisation
sept
8 nont installer() {
9     tandis que (! strip.begin());           // juge si l'initialisation a réussi
dix
11    strip.setAllLedsColor (0xFF0000); // Définit toutes les couleurs des LED sur rouge
12    retard (2000);
13    strip.setAllLedsColor (0x00FF00); // définit toutes les couleurs des LED sur le vert
14    retard (2000);
15    strip.setAllLedsColor (0x0000FF); // définit toutes les couleurs des LED sur bleu
16    retard (2000);
17    strip.setAllLedsColor (255, 255, 0); // définit toutes les couleurs des LED sur le jaune. Il s'agit simplement d'une forme différente de valeur RVB.
18
19    retard (2000);
20    strip.setAllLedsColor (0, 0, 0);          // éteint toutes les LED.
21    retard (2000);
22
23    strip.setLedColor (0, 255, 0, 0); retard (1000); // régler la LED N0.0 sur rouge
24
25    strip.setLedColor (1, 0, 255, 0); retard (1000); // met la LED N0.1 au vert
26
27    strip.setLedColor (2, 0, 0, 255); retard (1000); // régler la LED N0.2 sur bleu
28
29    strip.setLedColor (3, 255, 255, 0); // régler la LED N0.3 sur jaune
30    retard (1000);
31    strip.setLedColor (4, 255, 0, 255); // régler la LED N0.4 sur violet
32    retard (1000);
33 }
34 }
```

```

35 néant boucle() {
36     pour ( int k = 0; k <255; k = k + 2) {
37         strip.setAllLedsColorData (strip.Wheel (k)); // définir les données de couleur pour toutes les LED
38         strip.show (); // affiche le jeu de couleurs avant
39         retard (50);
40     }
41     retard (3000);
42
43     pour ( int j = 0; j <255; j += 2) {
44         pour ( int i = 0; je < LEDS_COUNT ; i++) {
45             strip.setLedColorData (i, strip.Wheel (i * 256 / LEDS_COUNT + j)); // définir les données de couleur pour la LED une par une
46
47         }
48         strip.show (); // affiche le jeu de couleurs
49         retard (50);
50     }
51 }
```

Tout d'abord, vous devez installer et inclure la bibliothèque.

comprendre "Freenove_WS2812B_RGBLED_Controller.h"

Puis réglez le nombre de LED. Le maximum est de 255. Mais il est limité par l'alimentation.

définir LEDS_COUNT dix // il définit le nombre de LED

Il existe deux façons de définir la couleur des LED.

UNE, régler la couleur de la LED directement.

1) définir toutes les couleurs de LED

strip.setAllLedsColor (valeur RVB)

La valeur rgb doit être une valeur hexadécimale, comme strip.setAllLedsColor (0x0000FF).

strip.setAllLedsColor (rvalue, gvalue, bvalue)

Ceci est une autre forme, comme strip.setAllLedsColor (255, 255, 0).

Vous pouvez trouver plus de valeurs RVB et de couleurs ici https://www.rapidtables.com/web/color/RGB_Color.html

2) définir la couleur pour une LED

strip.setLedColor (LED_Index, rgb_value) ou strip.setLedColor (LED_Index, rvalue, gvalue, bvalue)

Le numéro de LED commence à 0, comme strip.setLedColor (0, 255, 0, 0).

B, Définissez d'abord les données de couleur. Ensuite, utilisez la fonction show. C'est un plus efficace façon de définir les couleurs de nombreuses LED. Cela rendra le code plus rapide qu'auparavant. 1, régler toutes les couleurs de LED

strip.setAllLedsColorData (valeur rgb) ou strip.setAllLedsColorData (rvalue, gvalue, bvalue)

Après avoir terminé le réglage de la couleur, ajoutez simplement **strip.show ()**

2, définir la couleur pour une LED

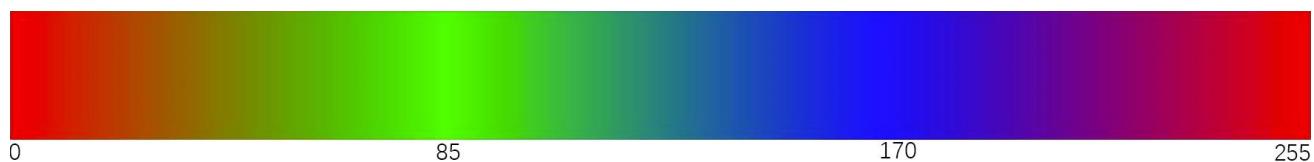
strip.setLedColorData (LED_Index, valeur rgb) ou strip.setLedColorData (LED_Index, rvalue, gvalue, bvalue)

Après avoir terminé le réglage de la couleur, ajoutez simplement **strip.show ()**

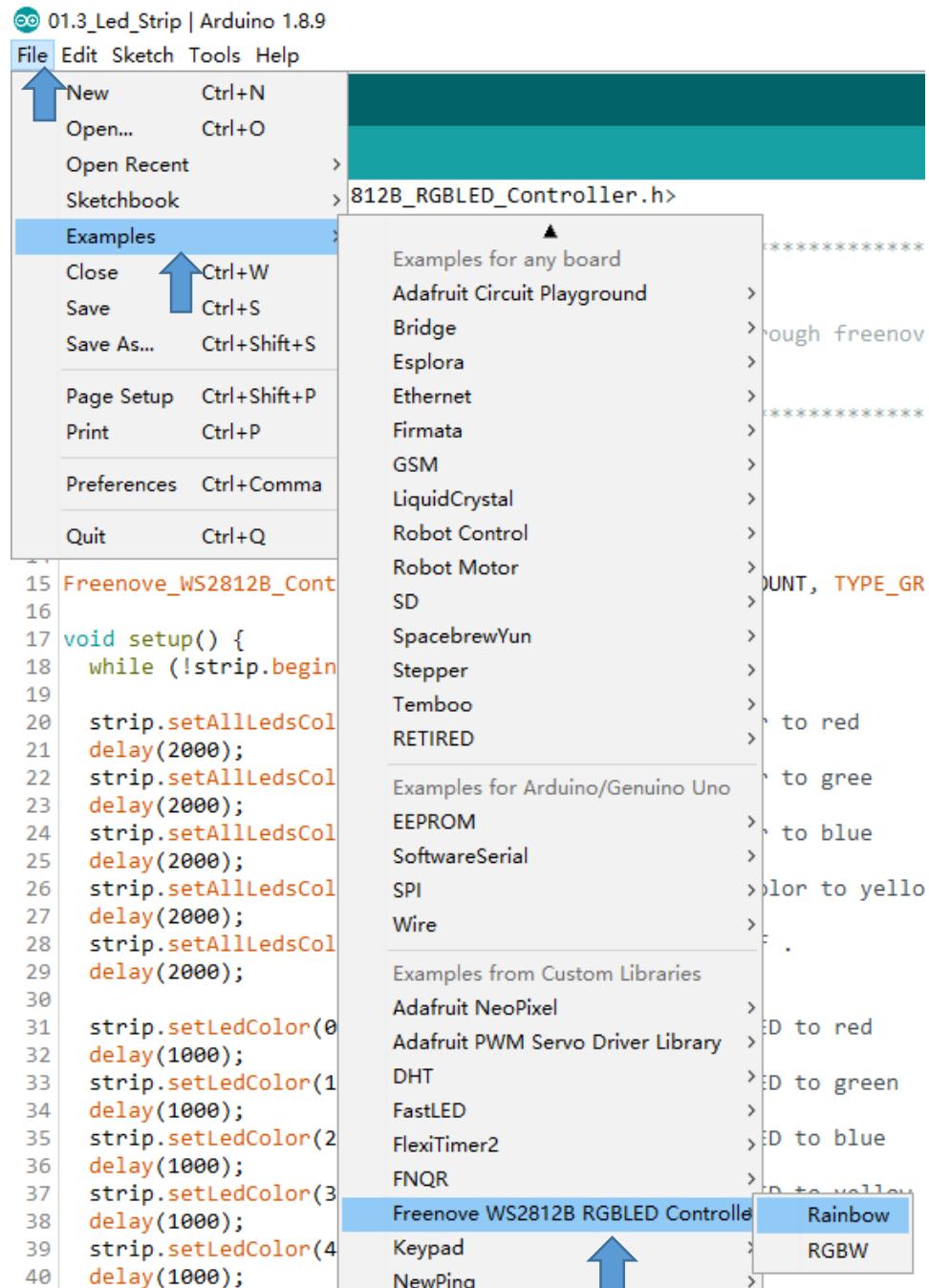
strip.Wheel (valeur) La valeur peut être comprise entre 0 et 255.

Des valeurs différentes correspondent à des couleurs différentes. Vous pouvez prendre **strip.Wheel (valeur)** comme valeur RVB à utiliser dans le

code, comme `strip.setAllLedsColorData (strip.Wheel (k))`.



Vous pouvez trouver quelques exemples de la bibliothèque ci-dessous:



1.4 Intégrer les fonctions

Nous allons maintenant écrire un code, qui comprend toutes les fonctions précédentes.

Code

01.4.1_Integrate_Functions

Vous devez d'abord supprimer le module de module Bluetooth lorsque vous téléchargez le code.

Téléchargez le code dans Sketches \ 01.4.1_Integrate_Functions. Ne séparez pas les fichiers du dossier.

ensuite placez votre voiture dans un endroit avec suffisamment d'espace pour se déplacer. Mettez la voiture sous tension.

```

1   # comprendre "Freenove_WS2812B_RGBLED_Controller.h"
2
3   # définir I2C_ADDRESS 0x20 // définir l'adresse I2C des LED
4   # définir LEDS_COUNT          dix      // il définit le nombre de LED
5
6   // définir les broches du moteur
7
8   # définir PIN_DIRECTION_LEFT 4
9   # définir PIN_DIRECTION_RIGHT 3
10  # définir PIN_MOTOR_PWM_LEFT 6
11  # définir PIN_MOTOR_PWM_RIGHT 5
12
13  # définir PIN_BATTERY          A0
14  # définir PIN_BUZZER           A0
15
16  # définir MOTOR_PWM_DEAD       dix
17
18  flotte batteryVoltage = 0;
19  booléen isBuzzered = faux ;
20
21
22  Bande Freenove_WS2812B_Controller ( I2C_ADDRESS , LEDS_COUNT , TYPE_GRB); // initialisation
23
24
25  néant installer() {
26    tandis que (! strip.begin()); // juge si l'initialisation réussit
27
28    Serial.begin (9600); // définir le débit en bauds
29    pinsSetup ();          // définir le mode broche (sortie ou entrée)
30
31    strip.setAllLedsColor (0x00FF00); // définit toutes les couleurs des LED sur le vert
32    motorRun (100, 100);        // La voiture avance
33    alarme (4, 1);            // 4 temps, 1 répétition
34    resetCarAction ();         // Arrêtez la voiture et éteignez le buzzer
35
36    strip.setAllLedsColor (0xFF0000); // Définit toutes les couleurs des LED sur rouge
37    motorRun (-100, -100); // La voiture recule
38    setBuzzer ( vrai );        // activer le buzzer

```

```
33     retard (1000);
34     resetCarAction ();           // Arrêtez la voiture et éteignez le buzzer
35 }
36
37 néant boucle() {
38     si (getBatteryVoltage () == vrai ) {Serial.print ( "tension:
39         ");
40     Serial.println (batteryVoltage);
41 }
42
43     int r = aléatoire (0, 255);      // une valeur aléatoire entre 0 ~ 255 // une
44     int g = aléatoire (0, 255);      valeur aléatoire entre 0 ~ 255 // une valeur
45     int b = aléatoire (0, 255);      aléatoire entre 0 ~ 255
46     strip.setAllLedsColor (r,        g, b); // Définit toutes les couleurs de LED avec la valeur actuelle r, g, b
47     retard (1000);
48 }
49
50 néant pinsSetup () {
51     pinMode ( PIN_DIRECTION_LEFT , PRODUCTION);
52     pinMode ( PIN_MOTOR_PWM_LEFT , PRODUCTION);
53     pinMode ( PIN_DIRECTION_RIGHT , PRODUCTION);
54     pinMode ( PIN_MOTOR_PWM_RIGHT , PRODUCTION);
55     setBuzzer ( faux ); // Désactiver le buzzer
56 }
57
58 néant motorRun ( int speedl , int speedr ) {
59     int dirL = 0, dirR = 0;
60     si ( speedl > 0 ) {dirL = 0;
61
62     }
63     autre {
64         dirL = 1;
65         speedl = - speedl ;
66     }
67
68     si ( speedr > 0 ) {dirR = 1;
69
70     }
71     autre {
72         dirR = 0;
73         speedr = - speedr ;
74     }
75
76     speedl = contraindre ( speedl , 0, 255);
```

```
77     speedr = contraindre ( speedr , 0, 255);  
78  
79     si (abdos( speedl ) < MOTOR_PWM_DEAD && abdos( speedr ) < MOTOR_PWM_DEAD ) {  
80         speedl = 0;  
81         speedr = 0;  
82     }  
83     digitalWrite ( PIN_DIRECTION_LEFT , dirL);  
84     digitalWrite ( PIN_DIRECTION_RIGHT , dirR);  
85     analogWrite ( PIN_MOTOR_PWM_LEFT , speedl );  
86     analogWrite ( PIN_MOTOR_PWM_RIGHT , speedr );  
87 }  
88  
89     booléen getBatteryVoltage () {  
90         si (! isBuzzered) {  
91             pinMode ( PIN_BATTERY , CONTRIBUTION);  
92             int batteryADC = analogRead ( PIN_BATTERY );  
93             si (batterieADC <614)           // 3 V / 12 V, tension de lecture: <2,1 V / 8,4 V  
94             {  
95                 batteryVoltage = batteryADC / 1023.0 * 5.0 * 4;  
96                 retourne vrai ;  
97             }  
98         }  
99         retourner faux ;  
100    }  
101  
102    néant setBuzzer ( booléen drapeau )  
103    {isBuzzered = drapeau ;  
104    pinMode ( PIN_BUZZER , drapeau );  
105    digitalWrite ( PIN_BUZZER , drapeau );  
106    }  
107  
108    néant alarme (u8 battre , u8 répéter ) {  
109        battre = contraindre ( battre , 1, 9);  
110        répéter = contraindre ( répéter , 1, 255);  
111        pour ( int j = 0; j < répéter ; j ++){  
112            pour ( int i = 0; je < battre ; i ++){  
113                setBuzzer ( vrai );  
114                retard (100);  
115                setBuzzer ( faux );  
116                retard (100);  
117            }  
118            retard (500);  
119        }  
120    }
```

```

121
122 néant resetCarAction () {
123     motorRun (0, 0);
124     setBuzzer ( faux );
125 }
```

01.4.2_Library_Integrate_Functions

Télécharger le code dans les esquisses \ 01.4.2_Library_Integrate_Functions, qui a la même fonction avec le code

01.4.1_Integrate_Functions. La seule différence est que nous écrivons certains codes dans la bibliothèque. Nous utiliserons cette bibliothèque plus tard.

Si les bibliothèques ne fonctionnent pas sur votre ordinateur, cela peut être lié à votre système informatique. Ne t'inquiète pas. Nous fournissons également un code pour les projets ultérieurs.

```

1 #include "Freenove_4WD_Car_for_Arduino.h"
2 #include "Freenove_WS2812B_RGBLED_Controller.h"
3
4 #define I2C_ADDRESS 0x20 //define I2C address of LEDs
5 #define LEDS_COUNT    10   //it defines number of LEDs
6
7 Freenove_WS2812B_Controller strip(I2C_ADDRESS, LEDS_COUNT, TYPE_GRB); //initializat
8
9 void setup() {
10     while (!strip.begin()); //judge if initialization successs
11 }
```

Le code principal est comme ci-dessous. Le Freenove_4WD_Car_for_Arduino.h a défini des broches, des variables et fonctions. Nous pouvons les utiliser directement. Nous n'avons pas besoin de les redéfinir.

```

1      # comprendre "Freenove_4WD_Car_for_Arduino.h"
2      # comprendre "Freenove_WS2812B_RGBLED_Controller.h"
3
4      # définir I2C_ADDRESS 0x20 // définir l'adresse I2C des LED
5      # définir LEDS_COUNT      dix      // il définit le nombre de LED
6      Bande Freenove_WS2812B_Controller ( I2C_ADDRESS , LEDS_COUNT , TYPE_GRB); // initialisation
sept
8      néant installer() {
9          tandis que (! strip.begin()); // juge si l'initialisation réussit
dix
10         Serial.begin (9600); // définir le débit en bauds
11         pinsSetup ();           // définir le mode broche (sortie ou entrée)
12
13         strip.setAllLedsColor (0x00FF00); // définit toutes les couleurs des LED sur le vert
14         motorRun (100, 100);           // La voiture avance
15         alarme (4, 1);              // 4 temps, 1 répétition
```

```

16   resetCarAction();           // Arrêtez la voiture et éteignez le buzzer
17
18   strip.setAllLedsColor (0xFF0000); // Définit toutes les couleurs des LED sur rouge
19   motorRun (-100, -100); // Voiture reculer
20   setBuzzer ( vrai );        // activer le buzzer
21   retard (1000);
22   resetCarAction ();         // Arrêtez la voiture et éteignez le buzzer
23 }
24
25 néant boucle() {
26   si ( getBatteryVoltage () == vrai ) {Serial.print ( "tension:
27     ");
28     Serial.println ( Voltage de batterie );
29   }
30   int r = aléatoire (0, 255);      // une valeur aléatoire entre 0 ~ 255 // une
31   int g = aléatoire (0, 255);      // valeur aléatoire entre 0 ~ 255 // une valeur
32   int b = aléatoire (0, 255);      // aléatoire entre 0 ~ 255
33   strip.setAllLedsColor (r,          g, b); // Définit toutes les couleurs de LED avec la valeur actuelle r, g, b
34   retard (1000);
35 }
```

Dans ce code, la bibliothèque Freenove_4WD_Car_for_Arduino est utilisée.

comprendre "Freenove_4WD_Car_for_Arduino.h"

Et vous pouvez le voir, nous n'avons pas défini les broches et les fonctions dans l'ensemble. Mais nous pouvons les utiliser.

Maintenant, apprenons une nouvelle compétence, en utilisant la bibliothèque.

Ouvrez le répertoire Sketches \ 01.4.2_Library_Integrate_Functions. Vous verrez les fichiers .cpp et .h. Ils sont **fichiers library**.

Lorsque vous utilisez la bibliothèque dans un nouveau code, il vous suffit de mettre ces deux fichiers dans le même répertoire de votre code.

 01.4.2_Library_Integrate_Functions.ino
 Freenove_4WD_Car_for_Arduino.cpp
 Freenove_4WD_Car_for_Arduino.h

Cette bibliothèque sera utilisée plus tard, ce qui rendra la programmation plus efficace. Nous allons maintenant introduire du contenu dans la bibliothèque.

Tout d'abord, vérifiez Freenove_4WD_Car_for_Arduino.h.

Ce fichier définit certaines broches et leur fonction.

Après avoir écrit le code # comprendre "Freenove_4WD_Car_for_Arduino.h"

Vous pourrez utiliser les broches et les fonctions définies dans Freenove_4WD_Car_for_Arduino.h. Et tu ne avoir besoin pour définir les broches et construire les fonctions plus dans votre code.

```

1 // Freenove_4WD_Car_for_Arduino.h
2 #ifndef _FREENOVE_4WD_CAR_FOR_ARDUINO_h
3 #define _FREENOVE_4WD_CAR_FOR_ARDUINO_h
```

```
4      # si défini (ARDUINO) && ARDUINO> = 100
5          # comprendre "arduino.h"
6      # autre
sept    # comprendre "WProgram.h"
8      # fin si
9      # définir PIN_SERVO           2
dix     // définir la broche du moteur
11     # définir PIN_DIRECTION_LEFT 4
12     # définir PIN_DIRECTION_RIGHT 3
13     # définir PIN_MOTOR_PWM_LEFT 6
14     # définir PIN_MOTOR_PWM_RIGHT 5
15
16     // définir la broche du module ultrasonique
17     # définir PIN SONIC TRIG      sept
18     # définir PIN SONIC ECHO      8
19
20     # définir PIN_IRREMOTE_RECV 9
21     // définir la broche SPI
22     # définir PIN_SPI_CE          9
23     # définir PIN_SPI_CSN         dix
24     # définir PIN_SPI_MOSI        11
25     # définir PIN_SPI_MISO        12
26     # définir PIN_SPI_SCK          13
27
28     # définir PIN_BATTERY         A0
29     # définir PIN_BUZZER          A0
30
31     // définir la broche du capteur de suivi
32     # définir PIN_TRACKING_LEFT  A1
33     # définir PIN_TRACKING_CENTER A2
34     # définir PIN_TRACKING_RIGHT A3
35     # définir MOTOR_PWM_DEAD      5
36
37     // définir des fonctions
38     // tout le contenu suivant est utilisé dans le code intégré ci-dessus
39     flotteur externe Voltage de batterie;
40     néant pinsSetup ();
41     néant motorRun ( int speedl , int speedr );
42     booléen getBatteryVoltage ();
43     néant setBuzzer ( booléen drapeau );
44     néant alarme (u8 battre , u8 répéter );
45     néant resetCarAction ();
46     # fin si
```

Puis vérifier **Freenove_4WD_Car_for_Arduino.cpp**.

Son nom doit être le même que celui du fichier .h. Nous définissons les fonctions dans le fichier .h et implémentons son contenu dans .cpp.

Tu peut voir comment la fonction fonctionne.

```
1 # comprendre "Freenove_4WD_Car_for_Arduino.h"
2 flotte batteryVoltage = 0;
3 booléen isBuzzered = faux ;
4
5 néant pinsSetup () {
6     // définir la broche du moteur
7     pinMode ( PIN_DIRECTION_LEFT , PRODUCTION);
8     pinMode ( PIN_MOTOR_PWM_LEFT , PRODUCTION);
9     pinMode ( PIN_DIRECTION_RIGHT , PRODUCTION);
dix    pinMode ( PIN_MOTOR_PWM_RIGHT , PRODUCTION);
11
12     // définir la broche du module ultrasonique
13     pinMode ( PIN SONIC_TRIG , PRODUCTION);
14     pinMode ( PIN SONIC_ECHO , CONTRIBUTION);
15
16     // définir la broche du capteur de suivi
17     pinMode ( PIN_TRACKING_LEFT , CONTRIBUTION);
18     pinMode ( PIN_TRACKING_RIGHT , CONTRIBUTION);
19     pinMode ( PIN_TRACKING_CENTER , CONTRIBUTION);
20     setBuzzer ( faux ); // désactiver le buzzer
21 }
22
23 néant motorRun ( int speedl , int speedr ) {
24     int dirL = 0, dirR = 0;
25     si ( speedl > 0) {dirL = 0;
26
27     }
autre {
    dirL = 1;
    speedl = - speedl ;
}
si ( speedr > 0) {dirR = 1;
}
autre {
    dirR = 0;
    speedr = - speedr ;
}
speedl = contraindre ( speedl , 0, 255); // la valeur absolue de speedl doit être comprise entre 0 et 255
speedr = contraindre ( speedr , 0, 255); // La valeur absolue de speedr doit être comprise entre 0 et 255
```

```
digitalWrite ( PIN_DIRECTION_LEFT , dirL);
digitalWrite ( PIN_DIRECTION_RIGHT , dirR);
analogWrite ( PIN_MOTOR_PWM_LEFT , speedl );
analogWrite ( PIN_MOTOR_PWM_RIGHT , speedr );
}

booléen getBatteryVoltage () {
    si (! isBuzzered) {
        pinMode ( PIN_BATTERY , CONTRIBUTION);
        int batteryADC = analogRead ( PIN_BATTERY );
        si (batterieADC <614)           // 3 V / 12 V, tension de lecture: <2,1 V / 8,4 V
        {
            batteryVoltage = batteryADC / 1023.0 * 5.0 * 4;
            retourne vrai ;
        }
    }
    retourner faux ;
}

néant setBuzzer ( booléen drapeau )
{
    isBuzzered = drapeau ;
    pinMode ( PIN_BUZZER , drapeau );
    digitalWrite ( PIN_BUZZER , drapeau );
}

néant alarme( u8 battre , u8 répéter ) {
    battre = contraindre ( battre , 1, 9);
    répéter = contraindre ( répéter , 1, 255);
    pour ( int j = 0; j < répéter ; j ++){
        pour ( int i = 0; je < battre ; i ++){
            setBuzzer ( vrai );
            retard (100);
            setBuzzer ( faux );
            retard (100);
        }
        retard (500);
    }
}

néant resetCarAction () {
    motorRun (0, 0);
    setBuzzer ( faux );
}
```

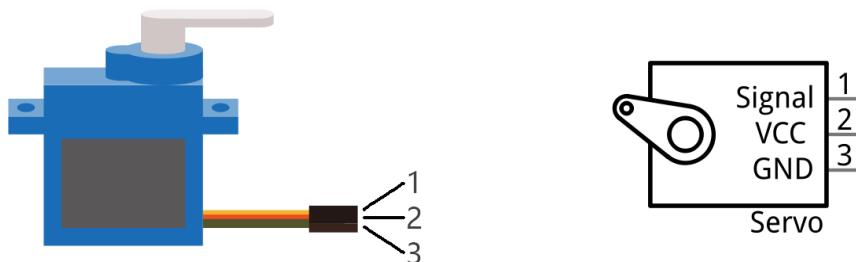

Chapitre 2 Évitement d'obstacles

Dans ce chapitre, nous présenterons le module servo et ultrasonique. Et puis faire une voiture qui peut automatiquement les obstacles.

Si vous avez des préoccupations, n'hésitez pas à nous contacter via support@freenove.com

2.1 Servo

Servo est un ensemble compact qui se compose d'un DCMotor, d'un ensemble d'engrenages réducteurs pour fournir le couple, d'un capteur et d'une carte de circuit imprimé de commande. La plupart des servos n'ont qu'une amplitude de mouvement de 180 degrés via leur « cornet ». Les servos peuvent produire un couple plus élevé qu'un simple moteur à courant continu seul et ils sont largement utilisés pour contrôler le mouvement dans les voitures miniatures, les modèles réduits d'avions, les robots, etc. Les servos ont trois fils qui se terminent généralement par une prise mâle ou femelle à 3 broches. Deux fils sont destinés à l'alimentation électrique: positif (2-VCC, fil rouge), négatif (3-GND, fil marron) et la ligne de signal (1 signal, fil orange) comme représenté dans le servo fourni dans votre kit.



Nous utiliserons un signal PWM 50Hz avec un rapport cyclique dans une certaine plage pour piloter le servo. Le temps durable 0.5ms-2,5 ms de niveau élevé de cycle unique PWM correspondent à l'angle d'asservissement de 0 degrés à 180 degrés linéairement. Une partie des valeurs correspondantes est la suivante:

Temps de haut niveau	Angle de servo
0,5 ms	0 degré
1 ms	45 degrés
1,5 ms	90 degrés
2 ms	135 degrés
2,5 ms	180 degrés

Lorsque vous changez le signal d'asservissement, le servo tournera jusqu'à la position désignée.

Code

02.1_Servo

Vous devez d'abord supprimer le module de module Bluetooth lorsque vous téléchargez le code.

Téléchargez le code dans Sketches \ 02.1_Servo. Ensuite, allumez la voiture. Ensuite, vous verrez le balayage des servos entre 45 °, 90 ° et 135 °. Le code est ci-dessous:

```
1 # comprendre "Servo.h"
```

```
// bibliothèque servo
```

```

2
3 # définir PIN_SERVO 2           // définir la broche servo
4
5 Servo;                      // crée un objet servo pour contrôler un servo
6 char servoOffset = 0;        // change la valeur en Calibrate servo
sept
8 néant installer() {
9   servo.attach ( PIN_SERVO );      // initialiser le servo
dix
10  servo.write (90 + servoOffset); // Calibrer le servo
11 }
12
13 néant boucle() {
14   servo.write (45);            // faire tourner le servo à 45 ° // retarder
15   retard (1000);             1000ms
16
17   servo.write (90);           // faire tourner le servo à 90 °
18   retard (1000);
19
20   servo.write (135);          // faire tourner le servo à 135 °
21   retard (1000);
22
23   servo.write (90);           // faire tourner le servo à 90 °
24   retard (1000);
25 }
26

```

Bibliothèque servo

Il y a quelques fonctions de cette bibliothèque:

servo.attach (broche)

Attachez la variable Servo à une broche.

pin: broche servo c'est une étape d'initialisation pour contrôler un servo.

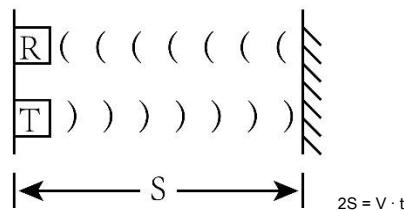
servo.write (angle)

Écrit une valeur sur le servo, contrôlant l'arbre en conséquence. angle: la valeur à écrire sur le servo, de 0 à 180

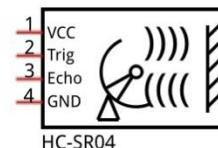
Pour plus de détails, veuillez consulter: <https://www.arduino.cc/en/Reference/Servo>

2.2 Module de télémétrie ultrasonique

Le module de télémétrie ultrasonique utilise le principe selon lequel les ondes ultrasonores se reflètent lorsqu'elles rencontrent des obstacles. Ceci est possible en comptant l'intervalle de temps entre le moment où l'onde ultrasonore est transmise et le moment où l'onde ultrasonore se réfléchit après avoir rencontré un obstacle. Le comptage de l'intervalle de temps se terminera après la réception d'une onde ultrasonore et la différence de temps (Δt) est le temps total du trajet de l'onde ultrasonore entre sa transmission et sa réception. Parce que la vitesse du son dans l'air est une constante et qu'elle est d'environ $v = 340 \text{ m / s}$, nous pouvons calculer la distance entre le module de télémétrie ultrasonique et l'obstacle: $s = vt / 2$.



Le module de télémétrie à ultrasons HC-SR04 intègre à la fois un émetteur à ultrasons et un récepteur. L'émetteur est utilisé pour convertir les signaux électriques (énergie électrique) en ondes sonores à haute fréquence (au-delà de l'audition humaine) (énergie mécanique) et la fonction du récepteur est à l'opposé. L'image et le schéma du module de télémétrie à ultrasons HC SR04 sont présentés ci-dessous:

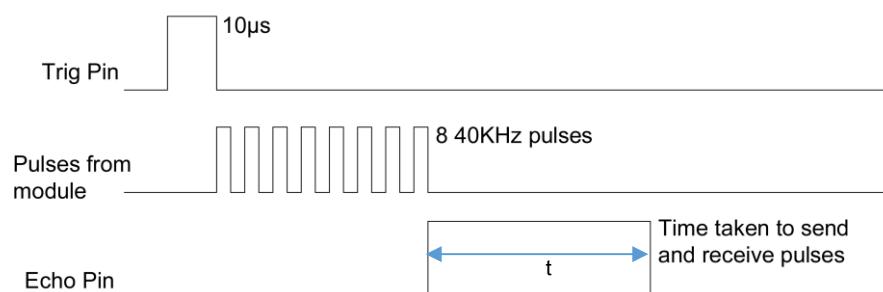


Description de la broche:

Nom de la broche	Code PIN	La description
Vcc	1	Électrode positive d'alimentation, la tension est une broche Triger 5V
Trigonométrie	2	
Écho	3	Broche d'écho
Gnd	4	Électrode négative d'alimentation

Mode d'emploi:

Sortir une impulsion de haut niveau dans la broche de déclenchement pendant au moins 10 μs , le module commence à transmettre des ondes ultrasonores. En même temps, la broche Echo est tirée **vers le haut**. Lorsque le module reçoit les ondes ultrasonores renvoyées après avoir rencontré un obstacle, la broche Echo sera tirée **vers le bas**. La durée du niveau haut dans la broche Echo est le temps total de l'onde ultrasonore de la transmission à la réception, $s = vt / 2$. Cela se fait constamment.



Code

02.2_Ultrasonic_Ranging

Ce projet est utilisé pour définir différents angles d'asservissement et détecter la distance d'obstacle à chaque angle.

Vous devez d'abord supprimer le module de module Bluetooth lorsque vous téléchargez le code.

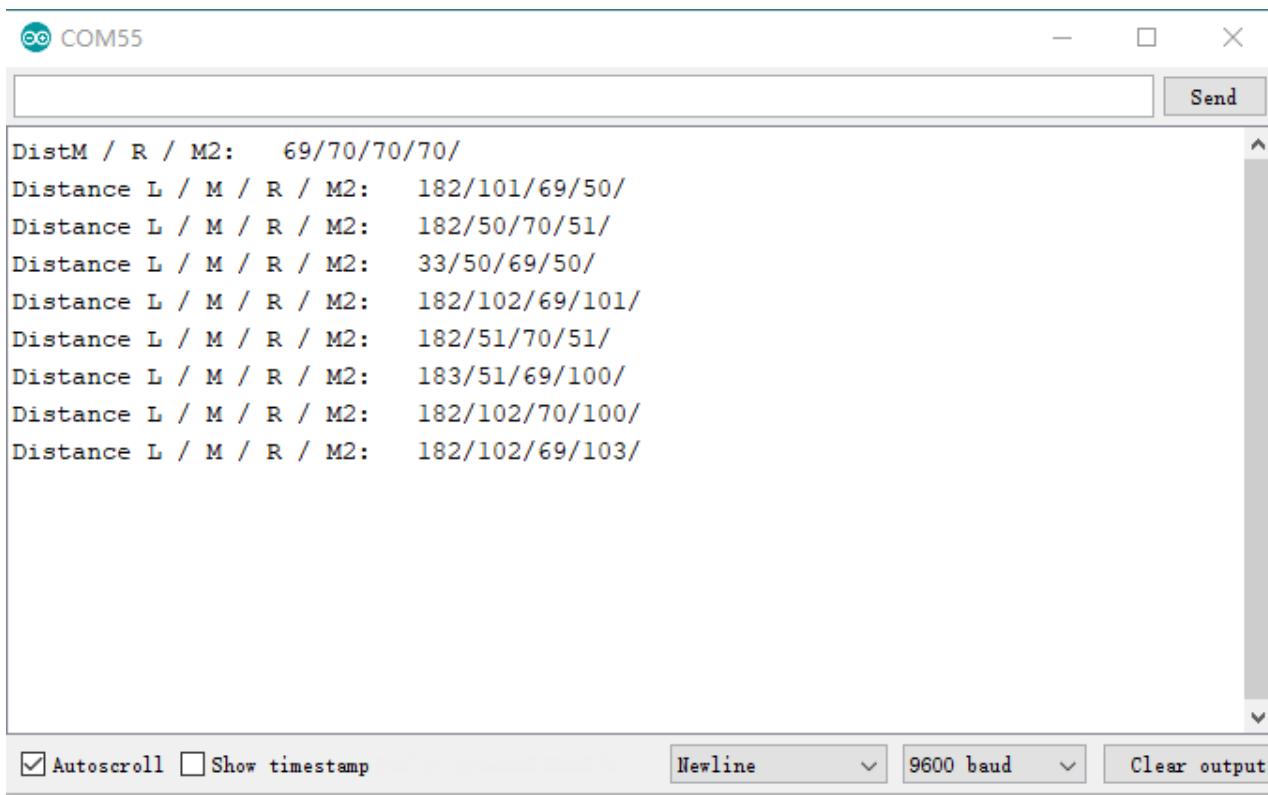
Téléchargez le code dans Sketches \ 02.2_Ultrasonic_Ranging.

Ensuite, allumez la voiture. Ouvrez ensuite le moniteur série.



```
02.2_Ultrasonic_Ranging | Arduino 1.8.9
File Edit Sketch Tools Help
02.2_Ultrasonic_Ranging §
1 //*****
2 * Product      : Freenove 4WD Car for UNO
3 * Description  : Ultrasonic ranging and servo.
4 * Author       : www.freenove.com
5 * Modification: 2019/08/05
6 ****
7 #include "Servo.h"           //include servo library
8
9 #define PIN_SERVO      2      //define servo pin
10
11#define PIN SONIC TRIG    7      //define Trig pin
12#define PIN SONIC ECHO    8      //define Echo pin
13
14#define MAX_DISTANCE    300    //cm
15#define COUNTS TIMEOUT   (MAX_DISTANCE*50) // calculate timeout
```

Ensuite, vous verrez le balayage des servos entre 45 °, 90 °, 135 °. Et le module à ultrasons détecte différentes distances à différents angles. Toutes les valeurs de distance sont imprimées ci-dessous:



Le code est ci-dessous:

```
1 # comprendre "Servo.h"                                // inclure la bibliothèque servo
2
3 # définir PIN_SERVO          2      // définir la broche servo
4
5 # définir PIN SONIC TRIG    sept   // définir la broche Trig
6 # définir PIN SONIC ECHO    8      // définir la broche Echo
sept
8 # définir MAX_DISTANCE      300    //cm
9 # définir SONIC_TIMEOUT     ( MAX_DISTANCE * 60 ) // calculer le timeout
dix
# définir SOUND_VELOCITY 340 // Vitesse du son: 340 m / s
11
12 Servo servo;           // créer un objet servo
13 char servoOffset = 0; u8 // change la valeur en Calibrate servo
14 distance [4];           // définir un tableau de type u8 (identique à un caractère non signé)
15
16 néant installer() {
17     Serial.begin (9600);
18     pinMode ( PIN SONIC TRIG , PRODUCTION); // définit trigPin en mode de sortie
19     pinMode ( PIN SONIC ECHO , CONTRIBUTION); // met echoPin en mode d'entrée
20     servo.attach ( PIN_SERVO );           // initialiser le servo
21     servo.write (90 + servoOffset); // changer servoOffset pour Calibrer le servo
22 }
23 néant boucle() {
```

```

24 servo.write (45);
25 retard (1000);
26 distance [0] = getSonar ();           // obtenir la valeur ultrasonique et la sauvegarder à distance [0]
27
28 servo.write (90);
29 retard (1000);
30 distance [1] = getSonar ();
31
32 servo.write (135);
33 retard (1000);
34 distance [2] = getSonar ();
35
36 servo.write (90);
37 retard (1000);
38 distance [3] = getSonar ();
39
40 Serial.print ( "Distance L / M / R / M2:          " ); // Gauche / Milieu / Droite / Milieu2
41 pour ( int i = 0; i <4; i ++ ) {
42     Serial.print (distance [i]);           // impression ultrasonique à 45 °, 90 °, 135 °, 90 °
43     Serial.print ( "/" );
44 }
45 Serial.print ( '\n' ); // le contenu suivant sera imprimé dans une nouvelle ligne
46 }
47
48 flotte getSonar () {
49     non signé longtemps pingTime;
50     flotte distance;
51     digitalWrite ( PIN SONIC TRIG , HAUTE); // rend la sortie trigPin de haut niveau pendant 10 µs pour triger HC_SR04,
52
53     delayMicrosecondes (10);
54     digitalWrite ( PIN SONIC TRIG , FAIBLE);
55     pingTime = pulseIn ( PIN SONIC ECHO , HAUTE, SONIC_TIMEOUT ); // Attendre le retour du HC-SR04 au niveau haut et mesurer
56     ce temps d'attente
57     si (pingTime! = 0)
58         distance = ( flotte ) pingTime * SOUND_VELOCITY / 2/10 000; // calculer la distance en fonction du temps
59
60     autre
61     distance = MAX_DISTANCE ;
62     revenir distance; // retourne la valeur de la distance
63 }
```

```

# définir MAX_DISTANCE      300    //cm
# définir SONIC_TIMEOUT      ( MAX_DISTANCE * 60 ) // calculer le timeout
# définir SOUND_VELOCITY 340 // Vitesse du son: 340 m / s
```

Calculez d'abord t pour la distance maximale.

$$340 (t / 10000000) / 2 = \text{MAX_DISTANCE} / 100 \quad t = 58,8 * \text{MAX_DISTANCE}$$

(L'unité de t est μs)

Nous définissons le délai d'expiration sur $\text{MAX_DISTANCE} * 60$, un peu plus grand, car la distance maximale du module ultrasonique est beaucoup plus grande.

```
pingTime = pulseIn ( PIN_SONIC_ECHO , HAUTE, SONIC_TIMEOUT ); // Attendre que HC-SR04 revienne au niveau haut
```

Si le temps d'écho de haut niveau est supérieur à `SONIC_TIMEOUT`, il renverra 0. Puis `pingTime = 0`. `pulseIn` (broche, valeur, délai)

pin: le numéro de la broche Arduino sur laquelle vous souhaitez lire l'impulsion. valeur: type

d'impulsion à lire: soit HIGH soit LOW.

timeout (facultatif): le nombre de microsecondes à attendre que l'impulsion démarre; la valeur par défaut est une seconde. Pour plus de détails, veuillez consulter:

<https://www.arduino.cc/reference/en/language/functions/advanced-io/pulsein/>

Tableau

Un tableau est une collection de variables accessibles avec un numéro d'index. Définissez un tableau dans

Arduino.

Type de données Nom du tableau [Nombre d'éléments]

`u8 distance [3];` définir un tableau nommé `distance`, son type de données est `u8`, qui comporte 3 éléments. L'index commence à 0.

`u8 = caractère non signé` Sa plage est de 0 ~ 2⁸, à savoir 0 ~ 255. La plage de `u16` est comprise entre 0 et 65535.

Pour plus de détails sur la baie, veuillez consulter:

<https://www.arduino.cc/reference/en/language/variables/data-types/array/>

2.3 Voiture d'évitement d'obstacles automatique

Code

02.3.1_Automatic_Obstacle_Avoidance

Vous devez d'abord supprimer le module de module Bluetooth lorsque vous téléchargez le code.

Téléchargez le code dans Sketches \ 02.3.1_Automatic_Obstacle_Avoidance.

Déconnectez ensuite le câble USB. Allumez la voiture. Et mettez la voiture dans un endroit avec suffisamment d'espace pour se déplacer. Le code est ci-dessous:

```
1 # comprendre <Servo.h>
2 # définir PIN_SERVO          2
3
4 # définir PIN_DIRECTION_LEFT 4
5 # définir PIN_DIRECTION_RIGHT 3
6 # définir PIN_MOTOR_PWM_LEFT 6
7 # définir PIN_MOTOR_PWM_RIGHT 5
8
9 # définir PIN SONIC TRIG      sept
10 # définir PIN SONIC ECHO      8
11
12 # définir PIN_BATTERY         A0
13
14 # définir OBSTACLE_DISTANCE    40
15 # définir OBSTACLE_DISTANCE_LOW 15
16
17 # définir MAX_DISTANCE        300 //cm
18 # définir SONIC_TIMEOUT       ( MAX_DISTANCE * 60 )
19 # définir SOUND_VELOCITY       340 // SoundVelocity: 340m / s
20
21 Servo servo;
22 char servoOffset = 0;
23 int speedOffset; // batteryVoltageCompensationToSpeed
24
25 néant installer() {
26     pinMode ( PIN_DIRECTION_LEFT , PRODUCTION);
27     pinMode ( PIN_MOTOR_PWM_LEFT , PRODUCTION);
28     pinMode ( PIN_DIRECTION_RIGHT , PRODUCTION);
29     pinMode ( PIN_MOTOR_PWM_RIGHT , PRODUCTION);
30
31     pinMode ( PIN_SONIC_TRIG , PRODUCTION); // définit trigPin en mode de sortie
32     pinMode ( PIN_SONIC_ECHO , CONTRIBUTION); // met echoPin en mode d'entrée
33     servo.attach ( PIN_SERVO );
```

```
34     calculerVoltageCompensation ();
35 }
36
37 néant boucle() {
38     updateAutomaticObstacleAvoidance ();
39 }
40
41 néant updateAutomaticObstacleAvoidance () {
42     int distance [3], tempDistance [3] [5], sumDisntance;
43     statique u8 leftToRight = 0, servoAngle = 0, lastServoAngle = 0; // 
44     const u8 scanAngle [2] [3] = {{150, 90, 30}, {30, 90, 150}};
45
46     pour ( int i = 0; i <3; i ++) {
47
48         servoAngle = scanAngle [leftToRight] [i];
49         servo.write (servoAngle);
50         si (lastServoAngle! = servoAngle) {
51             retard (130);
52         }
53         lastServoAngle = servoAngle;
54         pour ( int j = 0; j <5; j ++){
55             tempDistance [i] [j] = getSonar ();
56             délaiMicrosecondes (2 * SONIC_TIMEOUT );
57             sumDisntance += tempDistance [i] [j];
58         }
59         si (leftToRight == 0) {
60             distance [i] = sommeDisntance / 5;
61         }
62         autre {
63             distance [2 - i] = sommeDisntance / 5;
64         }
65         sumDisntance = 0;
66     }
67     leftToRight = (leftToRight + 1)% 2;
68
69     si (distance [1] < OBSTACLE_DISTANCE ) { // Trop peu de distance devant
70         si (distance [0]> distance [2] && distance [0]> OBSTACLE_DISTANCE ) { // La distance à gauche est
71             plus grande que la bonne distance
72                 motorRun (- (150 + speedOffset), - (150 + speedOffset)); //Reculer
73                 retard (100);
74                 motorRun (- (150 + speedOffset), (150 + speedOffset));
75         }
76         sinon si (distance [0] <distance [2] && distance [2]> OBSTACLE_DISTANCE )
77         { // La distance droite est supérieure à la distance gauche
```

```
78     motorRun (- (150 + speedOffset), - (150 + speedOffset)); //Reculer
79     retard (100);
80     motorRun ((150 + speedOffset), - (150 + speedOffset));
81 }
82 autre {                                // Entre dans le coin mort, recule, puis tourne.
83     motorRun (- (150 + speedOffset), - (150 + speedOffset)); retard (100);
84
85     motorRun (- (150 + speedOffset), (150 + speedOffset));
86 }
87 }
88 autre {                                // Aucun obstacle à venir
89     si (distance [0] < OBSTACLE_DISTANCE_LOW ) {           // Obstacles sur le devant gauche.
90         motorRun (- (150 + speedOffset), - (150 + speedOffset)); //Reculer
91         retard (100);
92         motorRun ((180 + speedOffset), (50 + speedOffset));
93     }
94     sinon si (distance [2] < OBSTACLE_DISTANCE_LOW ) {           // Obstacles sur le front droit.
95         motorRun (- (150 + speedOffset), - (150 + speedOffset)); //Reculer
96         retard (100);
97         motorRun ((50 + speedOffset), (180 + speedOffset));
98     }
99     autre {                                // Croisière
100        motorRun ((80 + speedOffset), (80 + speedOffset));
101    }
102 }
103 }
104
105 flotte getSonar () {
106     non signé longtemps pingTime;
107     flotte distance;
108     digitalWrite ( PIN SONIC TRIG , HAUTE); // rend la sortie trigPin de haut niveau pendant 10 µs pour triger HC_SR04,
109
110     delayMicrosecondes (10);
111     digitalWrite ( PIN SONIC TRIG , FAIBLE);
112     pingTime = pulseIn ( PIN SONIC ECHO , HAUTE, SONIC_TIMEOUT ); // Attendre le retour du HC-SR04 au niveau haut et mesurer
113     ce temps d'attente
114     si (pingTime! = 0)
115         distance = ( flotte ) pingTime * SOUND_VELOCITY / 2/10 000; // calculer la distance en fonction du temps
116
117     autre
118         distance = MAX_DISTANCE ;
119     revenir distance; // retourne la valeur de la distance
120 }
121 }
```

```
122 néant calculerVoltageCompensation () {  
123     flotte voltageOffset = 8.4 - getBatteryVoltage (); speedOffset =  
124     tensionOffset * 20;  
125 }  
126  
127 néant motorRun ( int speedl , int speedr ) {  
128     int dirL = 0, dirR = 0;  
129     si ( speedl > 0 ) {dirL = 0;  
130  
131     }  
132     autre {  
133         dirL = 1;  
134         speedl = - speedl ;  
135     }  
136  
137     si ( speedr > 0 ) {dirR = 1;  
138  
139     }  
140     autre {  
141         dirR = 0;  
142         speedr = - speedr ;  
143     }  
144     digitalWrite ( PIN_DIRECTION_LEFT , dirL);  
145     digitalWrite ( PIN_DIRECTION_RIGHT , dirR);  
146     analogWrite ( PIN_MOTOR_PWM_LEFT , speedl );  
147     analogWrite ( PIN_MOTOR_PWM_RIGHT , speedr );  
148 }  
149  
150  
151     flotte getBatteryVoltage () {  
152         pinMode ( PIN_BATTERY , CONTRIBUTION);  
153         int batteryADC = analogRead ( PIN_BATTERY );  
154         flotte batteryVoltage = batteryADC / 1023.0 * 5.0 * 4;  
155         revenir Voltage de batterie;  
156 }
```

Nous utilisons analogWrite (PIN_MOTOR, valeur PWM) pour faire fonctionner le moteur. La valeur PWM représente différents la vitesse lorsque la tension de la batterie change. Nous définissons donc un speedOffset pour compenser la différence.

```
1     néant calculerVoltageCompensation () {  
2         flotte voltageOffset = 8.4 - getBatteryVoltage (); speedOffset =  
3         tensionOffset * 20;  
4     }
```

20 est la valeur testée. Vous pouvez essayer de tester quelle est la valeur la plus appropriée.

Nous avons besoin d'un servo de contrôle à 150 °, 90 °, 30 °, 90 °, 150 °...

Obtenez 5 valeurs de module ultrasonique (distance) sous un angle. Puis calculez la distance moyenne.

```

1 int distance [3], tempDistance [3] [5], sumDisntance;
2 statique u8 leftToRight = 0, servoAngle = 0, lastServoAngle = 0; // 
3 const u8 scanAngle [2] [3] = {{150, 90, 30}, {30, 90, 150}};
4 pour ( int i = 0; i <3; i ++) {
5
6     servoAngle = scanAngle [leftToRight] [i];
7     servo.write (servoAngle);
8     si (lastServoAngle! = servoAngle) {
9         retard (130);
10    }
11    lastServoAngle = servoAngle;
12    pour ( int j = 0; j <5; j ++){
13        tempDistance [i] [j] = getSonar ();
14        délaiMicrosecondes (2 * SONIC_TIMEOUT );
15        sumDisntance += tempDistance [i] [j];
16    }
17    si (leftToRight == 0) {
18        distance [i] = sommeDisntance / 5;
19    }
20    autre {
21        distance [2 - i] = sommeDisntance / 5;
22    }
23    sumDisntance = 0;
24 }
25 leftToRight = (leftToRight + 1)% 2;
26

```

ensuite faire réagir la voiture en fonction des distances ci-dessus. À propos de sa logique, veuillez vous référer à l'organigramme.

```

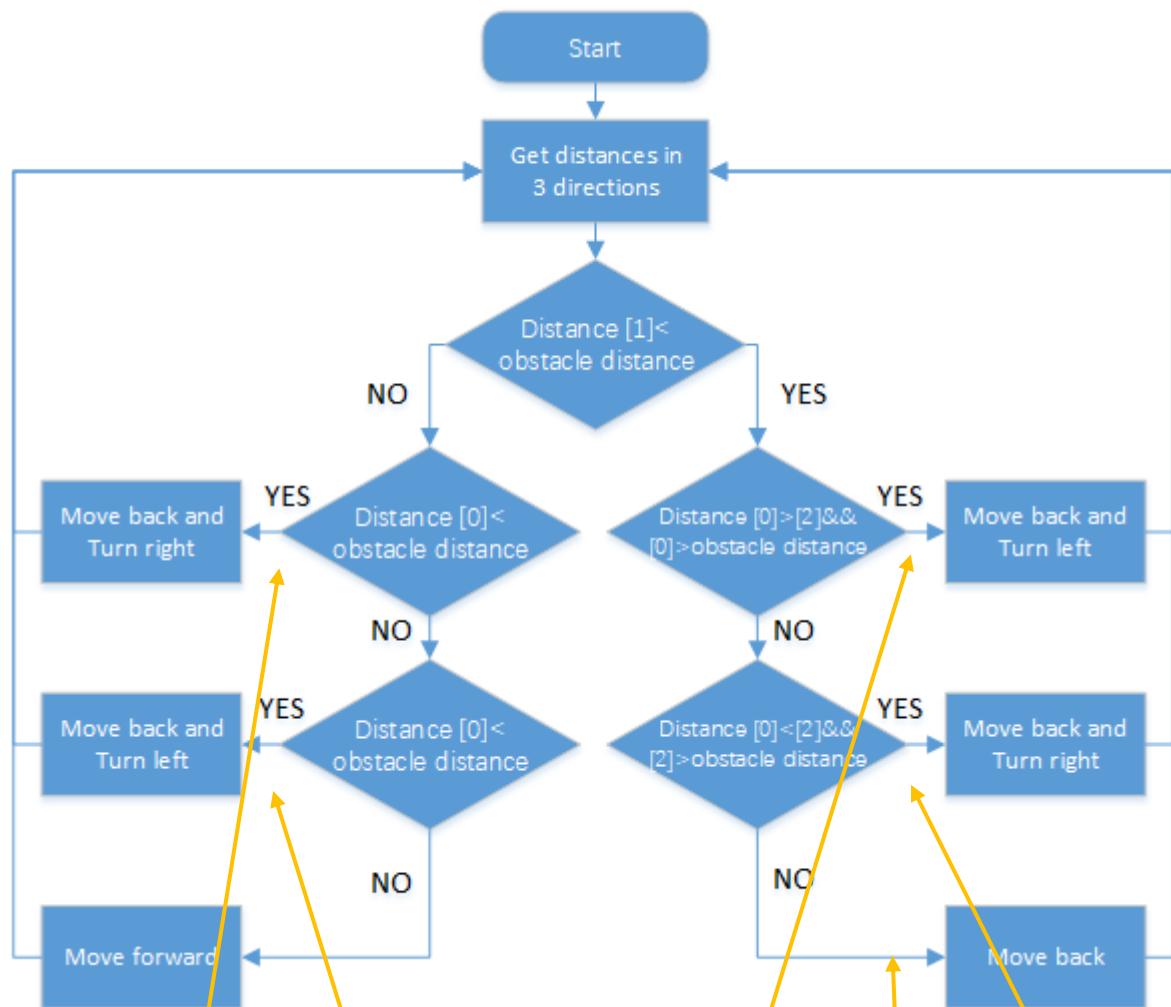
1 // faire réagir la voiture en fonction des distances
2 si (distance [1] < OBSTACLE_DISTANCE ) {                                // obstacle devant
3     si (distance [0]> distance [2] && distance [0]> OBSTACLE_DISTANCE ) {          // La distance à gauche est
4         plus grande que la bonne distance
5         motorRun (- (150 + speedOffset), - (150 + speedOffset)); // Recule un peu
6         retard (100);
7         motorRun (- (150 + speedOffset), (150 + speedOffset)); retard (50);
8     }
9 }
dix sinon si (distance [0] <distance [2] && distance [2]> OBSTACLE_DISTANCE )
11 {                               // La distance droite est supérieure à la distance gauche
12     motorRun (- (150 + speedOffset), - (150 + speedOffset)); // Recule un peu
13     retard (100);
14     motorRun ((150 + speedOffset), - (150 + speedOffset)); //Tournez à droite

```

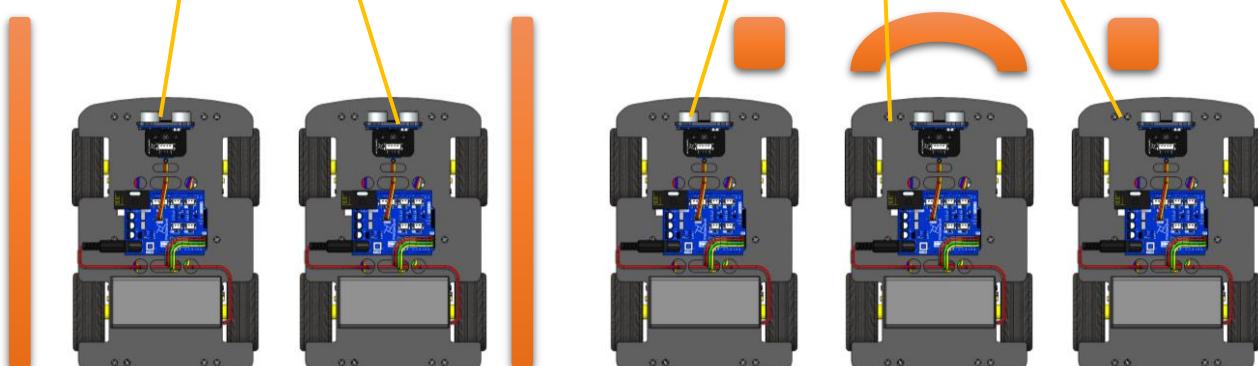
```
15     retard (50);
16 }
17 autre {                                // Entre dans le coin mort, recule un peu, puis tourne.
18     motorRun (- (150 + speedOffset), - (150 + speedOffset)); //Reculer
19     retard (100);
20     motorRun (- (150 + speedOffset), (150 + speedOffset)); retard (50);           //Tournez à gauche
21
22 }
23 }
24 autre {                                // Aucun obstacle à venir
25     si (distance [0] < OBSTACLE_DISTANCE_LOW ) {                           // Obstacles sur le devant gauche.
26         motorRun (- (150 + speedOffset), - (150 + speedOffset)); //Reculer
27         retard (100);
28         motorRun ((180 + speedOffset), (50 + speedOffset)); retard (50);       // Tourner à droite avec un grand rayon
29
30 }
31 sinon si (distance [2] < OBSTACLE_DISTANCE_LOW ) {motorRun (- (150 + speedOffset), su (150 + speedOffset)); //Obstacles sur le devant droit.
32     speedOffset); //Reculer
33     retard (100);
34     motorRun ((50 + speedOffset), (180 + speedOffset)); retard (50);           // Tourner à gauche avec un grand rayon
35
36 }
37 autre {                                // Croisière, aucun obstacle fermé
38     motorRun ((80 + speedOffset), (80 + speedOffset));
39 }
40 }
41 }
42 }
```

La logique du code est comme l'organigramme ci-dessous:

La distance [0] est la distance gauche de l'obstacle. La distance [1] est la distance du centre à l'obstacle. La distance [2] est la bonne distance par rapport à l'obstacle.



Ce projet envisage de suivre



02.3.2_Libery_Automatic_Obstacle_Avoidance

Le code dans les esquisses \ 02.3.2_Libery_Automatic_Obstacle_Avoidance comprend une bibliothèque

La fonction et le contenu du code sont similaires à ceux de la section précédente. Il y a peu de différence. Le code précédent est plus simple.

```

1 //*****
2 * Filename   : Automatic_Obstacle_Avoidance.ino
3 * Product    : Freenove 4WD Car for UNO
4 * Description : Automatic Obstacle Avoidance mode.
5 * Author     : www.freenove.com
6 * Modification: 2019/08/05
7 *****/
8
9 #include "Freenove_4WD_Car_for_Arduino.h"
10 #include "Automatic_Obstacle_Avoidance_Mode.h"
11
12 void setup() {
13     pinsSetup(); //from Freenove_4WD_Car_for_Arduino.h
14     servoSetup(); //from Automatic_Obstacle_Avoidance_Mode.h

```

Il y a 4 étiquettes.

Freenove_4WD_Car_for_Arduino.h a introduit dans la section [01.4.2_Library_Integrate_Functions](#)

L'étiquette principale est **02.3.2_Libery_Automatic_Obstacle_Avoidance.ino**

Son code est ci-dessous.

```

1 # comprendre "Freenove_4WD_Car_for_Arduino.h"
2 # comprendre "Automatic_Obstacle_Avoidance_Mode.h"
3
4 néant installer() {
5     pinsSetup(); // de Freenove_4WD_Car_for_Arduino.h
6     servoSetup(); // depuis Automatic_Obstacle_Avoidance_Mode.h
7     sept oa_CalculateVoltageCompensation(); //Automatic_Obstacle_Avoidance_Mode.h
8 }
9
dix néant boucle() {
10    updateAutomaticObstacleAvoidance(); //Automatic_Obstacle_Avoidance_Mode.h
11 }
12

```

Le code est très simple. La fonction d'autres étiquettes.

Code en étiquette Automatic_Obstacle_Avoidance_Mode.h

```

1 #ifndef _AUTOMATIC_OBSTACLE_AVOIDANCE_h
2 #define _AUTOMATIC_OBSTACLE_AVOIDANCE_h
3
4 # si défini (ARDUINO) && ARDUINO> = 100
5 # comprendre "arduino.h"
6 # autre
7 # comprendre "WProgram.h"
8 # fin si
9 # comprendre <EEPROM.h>
dix # comprendre "Servo.h"
11 # comprendre "Freenove_4WD_Car_for_Arduino.h"

```

```

12
13 # définir OA_SERVO_CENTER (90)
14 # définir OA_SCAN_ANGLE_INTERVAL 50
15 # définir OA_SCAN_ANGLE_MIN ( OA_SERVO_CENTER - OA_SCAN_ANGLE_INTERVAL )
16 # définir OA_SCAN_ANGLE_MAX ( OA_SERVO_CENTER + OA_SCAN_ANGLE_INTERVAL )
17 # définir OA_WAITTING_SERVO_TIME 130
18
19 # définir OA_CRUISE_SPEED (110 + oa_VoltageCompensationToSpeed)
20
21 # définir OA_ROTATY_SPEED_LOW (120 + oa_VoltageCompensationToSpeed)
22 # définir OA_ROTATY_SPEED_NORMAL (150 + oa_VoltageCompensationToSpeed)
23 # définir OA_ROTATY_SPEED_HIGH (180 + oa_VoltageCompensationToSpeed)
24
25 # définir OA_TURN_SPEED_LV4 (180 + oa_VoltageCompensationToSpeed)
26 # définir OA_TURN_SPEED_LV1 (50 + oa_VoltageCompensationToSpeed)
27
28 # définir OA_BACK_SPEED_LOW (110 + oa_VoltageCompensationToSpeed)
29 # définir OA_BACK_SPEED_NORMAL (150 + oa_VoltageCompensationToSpeed)
30 # définir OA_BACK_SPEED_HIGH (180 + oa_VoltageCompensationToSpeed)
31
32 # définir OA_OBSTACLE_DISTANCE 40
33 # définir OA_OBSTACLE_DISTANCE_LOW 15
34
35 # définir OA_SPEED_OFFSET_PER_V 35
36
37 # définir OA_SERVO_OFFSET_ADDR_IN_EEPROM 0
38
39 # définir MAX_DISTANCE 300 //cm
40 # définir SONIC_TIMEOUT ( MAX_DISTANCE * 60 )
41 # définir SOUND_VELOCITY 340 // SoundVelocity: 340m / s
42
43
44 externe Servo servo;
45 externe int servoOffset;
46
47 néant servoSetup (); // initialise le servo
48 néant setServoOffset (entier décalage ); // Définir le décalage du servo
49 néant writeServo (u8 n ); // définir le servo sur un angle, le décalage a été pris en compte
50 néant writeServoOffsetToEEPROM ();
51 néant getServoOffsetFromEEPROM ();
52
53 flotte getSonar ();
54 néant oa_CalculateVoltageCompensation (); // Calculer la compensation de tension
55 néant updateAutomaticObstacleAvoidance ();

```

56	# fin si
----	----------

.h fichier utilisé pour définir les broches, les variables et les fonctions.

Vous pouvez obtenir des informations sur ce que vous pouvez utiliser à partir du fichier .h.

Le code d'implémentation détaillé des fonctions est inclus dans le fichier .cpp.

1 # comprendre "Automatic_Obstacle_Avoidance_Mode.h"	2
3 Servo servo;	4 char servoOffset = 0;
5 int oa_VoltageCompensationToSpeed;	6
7	sept néant servoSetup () {
8	getServoOffsetFromEEPROM ();
9	servo.attach (PIN_SERVO);
10 dix servo.write (90 + servoOffset);	11 }
12	13 néant setServoOffset (car décalage) {servoOffset = décalage = contraindre (décalage
14 , -100, 100); servo.write (90 + décalage);	15 }
16 }	17 néant writeServo (u8 n)
18 {	19 servo.write (90 + servoOffset);
20 }	21 }
22	23 néant writeServoOffsetToEEPROM () {
24 servo.write (90 + servoOffset);	25 EEPROM.write (OA_SERVO_OFFSET_ADDR_IN_EEPROM , servoOffset);
26 }	27
28 néant getServoOffsetFromEEPROM () {	29 servoOffset = EEPROM.read (OA_SERVO_OFFSET_ADDR_IN_EEPROM);
30 servoOffset = contrainte (servoOffset, -10, 10);	31 }
32	33 flotte getSonar () {
34 non signé longtemps pingTime;	35 flotte distance;
36 digitalWrite (PIN SONIC TRIG , HAUTE); // rend la sortie trigPin de haut niveau pendant 10 µs pour triger HC_SR04,	37 delayMicrosecondes (10);
38 digitalWrite (PIN SONIC TRIG , FAIBLE);	39 }

```
40     pingTime = pulseIn ( PIN SONIC_ECHO , HAUTE, SONIC_TIMEOUT ); // Attendre le retour du HC-SR04 au niveau haut et mesurer
41     ce temps d'attente
42
43     si (pingTime! = 0)
44
45         distance = ( flotte ) pingTime * SOUND_VELOCITY / 2/10 000; // calculer la distance en fonction du temps
46
47     autre
48         distance = MAX_DISTANCE ;
49
50     revenir distance; // retourne la valeur de la distance
51 }
52
53
54 néant oa_CalculateVoltageCompensation () {
55
56     getBatteryVoltage ();
57
58     flotte voltageOffset = BAT_VOL_STANDARD - Voltage de batterie; oa_VoltageCompensationToSpeed =
59     tensionOffset * OA_SPEED_OFFSET_PER_V ;
60 }
61
62
63 néant updateAutomaticObstacleAvoidance () {
64
65     int distance [3], tempDistance [3] [5], sumDisntance;
66
67     statique u8 cnt = 0, servoAngle = 0, lastServoAngle = 0; // 
68
69     si (cnt == 0) {
70
71         pour ( int i = 0; i <3; i ++ ) {
72
73             servoAngle = OA_SCAN_ANGLE_MAX - je * OA_SCAN_ANGLE_INTERVAL + servoOffset; servo.write
74             (servoAngle);
75
76             si (lastServoAngle! = servoAngle) {
77
78                 retard( OA_WAITTING_SERVO_TIME );
79
80             }
81
82             lastServoAngle = servoAngle;
83
84             pour ( int j = 0; j <5; j ++ ) {
85
86                 tempDistance [i] [j] = getSonar ();
87
88                 délaiMicrosecondes (2 * SONIC_TIMEOUT );
89
90                 sumDisntance += tempDistance [i] [j];
91
92             }
93
94             distance [i] = sommeDisntance / 5;
95
96             sumDisntance = 0;
97
98         }
99
100        cnt++;
101    }
102
103    autre {
104
105        pour ( int i = 2; i> 0; je-- ) {
106
107            servoAngle = OA_SCAN_ANGLE_MAX - je * OA_SCAN_ANGLE_INTERVAL + servoOffset; servo.write
108            (servoAngle);
109
110            si (lastServoAngle! = servoAngle) {
111
112                retard( OA_WAITTING_SERVO_TIME );
113
114            }
115
116        }
117
118    }
119
120 }
```

```

84         lastServoAngle = servoAngle;
85         pour ( int j = 0; j <5; j ++ ) {
86             tempDistance [i] [j] = getSonar ();
87             délaiMicrosecondes (2 * SONIC_TIMEOUT );
88             sumDisntance += tempDistance [i] [j];
89         }
90         distance [i] = sommeDisntance / 5;
91         sumDisntance = 0;
92     }
93     cnt = 0;
94 }
95
96     si (distance [1] < OA_OBSTACLE_DISTANCE ) { // Trop peu de distance devant
97         si (distance [0]> OA_OBSTACLE_DISTANCE || distance [2]> OA_OBSTACLE_DISTANCE ) {motorRun (- OA_BACK_SPEED_LOW
98             , - OA_BACK_SPEED_LOW ); // Recule un peu
99             retard (100);
100            si (distance [0]> distance [2]) { // La distance à gauche est supérieure à
101                bonne distance
102                    motorRun (- OA_ROTATY_SPEED_LOW , OA_ROTATY_SPEED_LOW );
103                }
104                autre { // La bonne distance est supérieure à
105                    distance gauche
106                        motorRun ( OA_ROTATY_SPEED_LOW , - OA_ROTATY_SPEED_LOW );
107                    }
108                }
109                autre { // Entre dans le coin mort, bouge
110                    reculer un peu, puis tourner.
111                        motorRun (- OA_BACK_SPEED_HIGH , - OA_BACK_SPEED_HIGH );
112                        retard (100);
113                        motorRun (- OA_ROTATY_SPEED_NORMAL , OA_ROTATY_SPEED_NORMAL );
114                    }
115                }
116 // le code suivant est différent de la dernière section. La dernière section est plus simple
117     autre { // Aucun obstacle à venir
118         si (distance [0] < OA_OBSTACLE_DISTANCE ) { // Obstacles sur le devant gauche.
119             si (distance [0] < OA_OBSTACLE_DISTANCE_LOW ) { // Très proche de l'avant gauche
120                 obstacle.
121                     motorRun (- OA_BACK_SPEED_LOW , - OA_BACK_SPEED_LOW ); //Reculer
122                     retard (100);
123                 }
124                 motorRun ( OA_TURN_SPEED_LV4 , OA_TURN_SPEED_LV1 );
125             }
126             sinon si (distance [2] < OA_OBSTACLE_DISTANCE ) { // Obstacles à droite
127                 de face.

```

```

128      si (distance [2] < OA_OBSTACLE_DISTANCE_LOW ){           // Très proche de la droite
129          obstacle avant.
130          motorRun (- OA_BACK_SPEED_LOW , - OA_BACK_SPEED_LOW );    //Reculer
131          retard (100);
132      }
133      motorRun ( OA_TURN_SPEED_LV1 , OA_TURN_SPEED_LV4 );
134  }
135  autre {                                         // Croisière
136      motorRun ( OA_CRUISE_SPEED , OA_CRUISE_SPEED );
137  }
138 }
139 }
```

Comme le servo peut ne pas être installé à 90 °, nous devons l'ajuster. Mais nous ne voulons pas l'ajuster à chaque fois, donc nous pouvons enregistrer le servoOffset dans un endroit et la prochaine fois, nous pouvons l'utiliser directement. Le microcontrôleur sur les cartes Arduino et Genuino a 1024 octets d'EEPROM.

C'est utile lorsque nous utilisons la télécommande pour calibrer le servo.

```

1 néant writeServoOffsetToEEPROM () {
2     servo.write (90 + servoOffset);
3     EEPROM.write ( OA_SERVO_OFFSET_ADDR_IN_EEPROM , servoOffset);
4 }
5
6 néant getServoOffsetFromEEPROM () {
sept     servoOffset = EEPROM.read ( OA_SERVO_OFFSET_ADDR_IN_EEPROM );
8     servoOffset = contrainte (servoOffset, -10, 10);
9 }
```

EEPROM

Le microcontrôleur des cartes Arduino et Genuino a 1024 octets d'EEPROM. EEPROM.write (adresse, valeur); écrire val à l'adresse

valeur = EEPROM.read (adresse); lire les données enregistrées dans l'adresse

par exemple,

```
EEPROM.write (0, 3);
EEPROM.write (1, 5);
EEPROM.write (2, 8);
```

Valeur0 = EEPROM.read (0); alors Value0 = 3 Value1 = EEPROM.read (2); then Value2 = 8 Pour plus d'informations, reportez-vous à <https://www.arduino.cc/en/Reference/EEPROM>

L'autre logique de code est similaire à la section précédente.

Chapitre 3 Suivi de ligne

Dans ce chapitre, nous présenterons le capteur de suivi de ligne et créerons une voiture de suivi de ligne.

Si vous avez des préoccupations, n'hésitez pas à nous contacter via support@freenove.com

3.1 Capteur de suivi de ligne

Il y a trois capteurs optiques réfléchissants sur cette voiture. Lorsque la lumière infrarouge émise par la diode infrarouge brille sur la surface de différents objets, le capteur recevra une lumière d'intensité différente après réflexion.

Comme nous le savons, les objets noirs absorbent mieux la lumière. Ainsi, lorsque des lignes noires sont dessinées sur le plan blanc, le capteur peut détecter la différence.

Le capteur peut également être appelé capteur de suivi de ligne.

Attention:

Le capteur optique réfléchissant (y compris le capteur de suivi de ligne) doit être évité dans un environnement avec des interférences infrarouges, comme la lumière du soleil.

La lumière du soleil contient beaucoup de lumière invisible comme l'infrarouge et l'ultraviolet. Dans un environnement avec une lumière solaire intense, le capteur optique réfléchissant ne peut pas fonctionner normalement.

Le tableau suivant montre les valeurs de tous les cas où trois capteurs de suivi détectent des objets de couleurs différentes. Parmi eux, des objets noirs ou aucun objet n'a été détecté pour représenter 1, et des objets blancs ont été détectés pour représenter 0.

La gauche	Milieu	Droite	Valeur (binaire)	Valeur (décimale)
0	0	0	000	0
0	0	1	001	1
0	1	0	010	2
0	1	1	011	3
1	0	0	100	4
1	0	1	101	5
1	1	0	110	6
1	1	1	111	sept

Code

03.1_Tracking_Sensor

Vous devez d'abord supprimer le module de module Bluetooth lorsque vous téléchargez le code.

Téléchargez le code dans Sketches \ 03.1_Tracking_Sensor, mettez l'appareil sous tension et ouvrez le moniteur série. Mettez quelque chose de noir sous le capteur. Déplacez-le et vous verrez différentes LED s'allumer.



Le moniteur est illustré ci-dessous:

```

Sensor Value (L / M / R / ALL) : 0      1      1      3
Sensor Value (L / M / R / ALL) : 0      0      1      1
Sensor Value (L / M / R / ALL) : 1      0      0      4
Sensor Value (L / M / R / ALL) : 1      0      0      4
Sensor Value (L / M / R / ALL) : 0      0      1      1
Sensor Value (L / M / R / ALL) : 0      0      1      1
Sensor Value (L / M / R / ALL) : 0      0      1      1
Sensor Value (L / M / R / ALL) : 0      0      1      1
Sensor Value (L / M / R / ALL) : 0      0      1      1
Sensor Value (L / M / R / ALL) : 0      0      1      1
Sensor Value (L / M / R / ALL) : 0      0      1      1
Sensor Value (L / M / R / ALL) : 0      0      1      1
Sensor Value (L / M / R / ALL) : 0      0      1      1
Sensor Value (L / M / R / ALL) : 0      0      1      1
Sensor Value (L / M / R / ALL) : 0      0      1      1
Sensor Value (L / M / R / ALL) : 0      0      1      1

```

Autoscroll Show timestamp Newline 9600 baud Clear output

Le code est ci-dessous:

```

1 # définir PIN_TRACKING_LEFT          A1
2 # définir PIN_TRACKING_CENTER       A2
3 # définir PIN_TRACKING_RIGHT        A3
4
5 u8 sensorValue [4]; // définir un tableau, u8 = caractère non signé, 0 ~ 255.
6
7 sept néant installer() {
8     Serial.begin (9600); // définir le débit en bauds
9     pinMode ( PIN_TRACKING_LEFT , CONTRIBUTION); //
dix    pinMode ( PIN_TRACKING_RIGHT , CONTRIBUTION); //
11    pinMode ( PIN_TRACKING_CENTER , CONTRIBUTION); //
12 }
13

```

```
14 néant boucle() {  
15     sensorValue [0] = digitalRead ( PIN_TRACKING_LEFT );  
16     sensorValue [1] = digitalRead ( PIN_TRACKING_CENTER );  
17     sensorValue [2] = digitalRead ( PIN_TRACKING_RIGHT );  
18     sensorValue [3] = sensorValue [0] << 2 | sensorValue [1] << 1 | sensorValue [2]; Serial.print ( "Valeur du capteur  
19     (L / M / R / ALL):" );  
20     pour ( int i = 0; i < 4; i ++ ) {  
21         Serial.print (sensorValue [i]);  
22         Serial.print ( '\t' ); // signifie Tab  
23     }  
24     Serial.print ( '\n' );           // signifie nouvelle ligne  
25     retard (500);  
26 }  
27 }
```

Opérateurs au niveau du bit

Il existe des opérateurs au niveau du bit. << (décalage de bits vers la gauche)

Si `sensorValue [0] = 1, sensorValue [1] = 1, sensorValue [2] = 1` `sensorValue [0] << 2`
then `sensorValue [0] = 100`, (Binary), soit 4 (Decimal)

`sensorValue [1] << 1` then `sensorValue [0] = 010`, (Binary), soit 2 (Decimal)

`sensorValue [2] = 001` (binaire)
| (bit à bit ou)

Le code se transforme en: `100 | 010 | 001 = 111` (binaire), soit 7 (décimal)

>> (décalage vers la droite)

& (bit à bit et)

^ (xor au niveau du bit)

~ (pas au niveau du bit)

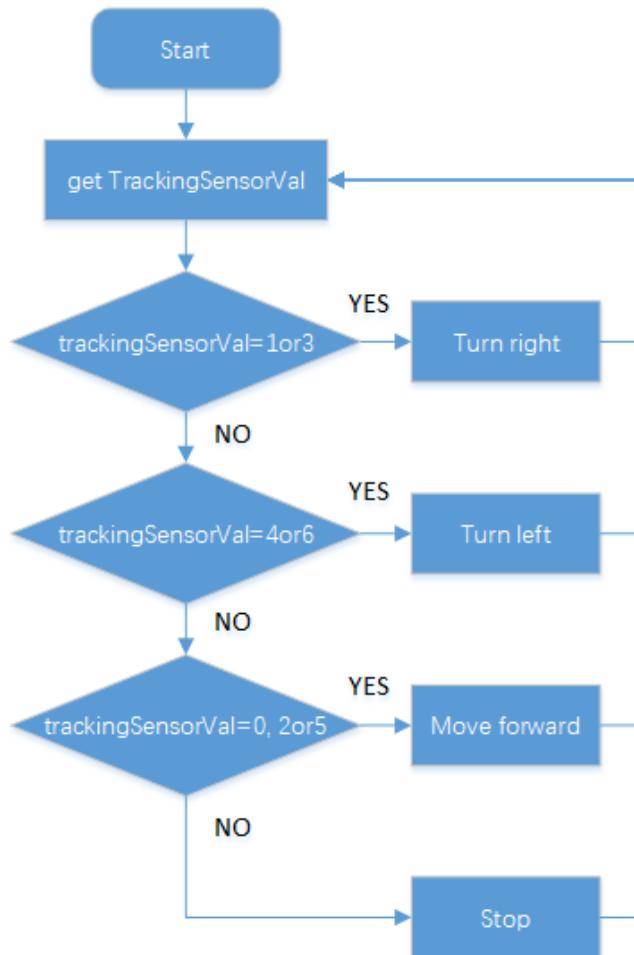
Pour plus d'informations, veuillez consulter: <https://www.arduino.cc/reference/>

3.2 Voiture de suivi de ligne

La voiture effectuera différentes actions en fonction de la valeur transmise par le capteur de suivi de ligne. Quand

La gauche	Milieu	Droite	Valeur (binaire)	Valeur (décimale)	action
0	0	0	000	0	Avance
0	0	1	001	1	Tournez à droite
0	1	0	010	2	Avance
0	1	1	011	3	Tournez à droite
1	0	0	100	4	Tourner à gauche
1	0	1	101	5	Avance
1	1	0	110	6	Tourner à gauche
1	1	1	111	sept	Arrêtez

L'organigramme de la voiture de suivi de ligne est comme ci-dessous:

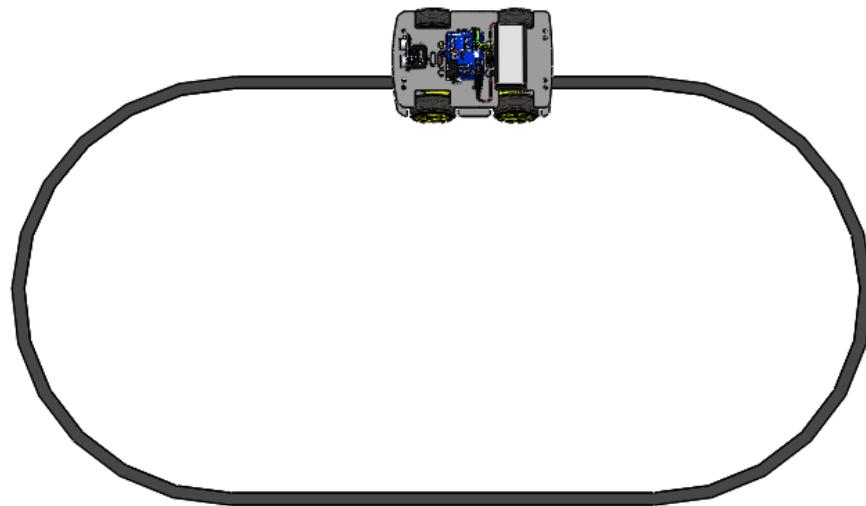


Code

03.2_Automatic_Tracking_Line

Vous devez d'abord supprimer le module de module Bluetooth lorsque vous téléchargez le code.

Télécharger le code dans **Croquis \ 03.3_One_Code_Automatic_Tracking_Line**. Allumer l'appareil. Utilisez un ruban noir pour construire une ligne, puis placez votre voiture dessus comme ci-dessous.



Vous pouvez également choisir 03.2_Automatic_Tracking_Line. Le code est le même. Le code est

ci-dessous.

```

1  # comprendre "Freenove_4WD_Car_for_Arduino.h"
2
3  # définir TK_STOP_SPEED          0
4  # définir TK_FORWARD_SPEED      (90 + tk_VoltageCompensationToSpeed) )
5
6  // définir différents niveaux de vitesse
7  # définir TK_TURN_SPEED_LV4      (180 + tk_VoltageCompensationToSpeed) )
8  # définir TK_TURN_SPEED_LV3      (150 + tk_VoltageCompensationToSpeed) )
9  # définir TK_TURN_SPEED_LV2      (-140 + tk_VoltageCompensationToSpeed) (-160 +
10 # définir TK_TURN_SPEED_LV1      tk_VoltageCompensationToSpeed)
11
12 int tk_VoltageCompensationToSpeed; // définir la compensation de vitesse de tension
13
14 néant installer() {
15     pinsSetup(); // configurer les broches, à partir de la bibliothèque Freenove_4WD_Car_for_Arduino.h
16     getTrackingSensorVal(); // Calculer la compensation de vitesse de tension
17 }
18
19 néant boucle() {
20     u8 trackingSensorVal = 0;
21     trackingSensorVal = getTrackingSensorVal(); // obtenir la valeur du capteur
22 }
```

```
23     commutateur (trackingSensorVal)
24     {
25         Cas 0:      // 000
26             motorRun ( TK_FORWARD_SPEED , TK_FORWARD_SPEED ); // voiture avance
27             Pause ;
28         Cas sept:    // 111
29             motorRun ( TK_STOP_SPEED , TK_STOP_SPEED ); // arrêt de voiture
30             Pause ;
31         Cas 1:      // 001
32             motorRun ( TK_TURN_SPEED_LV4 , TK_TURN_SPEED_LV1 ); // tour de voiture
33             Pause ;
34         Cas 3:      // 011
35             motorRun ( TK_TURN_SPEED_LV3 , TK_TURN_SPEED_LV2 ); // voiture tourner à droite
36             Pause ;
37         Cas 2:      // 010
38         Cas 5:      // 101
39             motorRun ( TK_FORWARD_SPEED , TK_FORWARD_SPEED ); // voiture avance
40             Pause ;
41         Cas 6:      // 110
42             motorRun ( TK_TURN_SPEED_LV2 , TK_TURN_SPEED_LV3 ); // voiture tourner à gauche
43             Pause ;
44         Cas 4:      // 100
45             motorRun ( TK_TURN_SPEED_LV1 , TK_TURN_SPEED_LV4 ); // voiture tourner à droite
46             Pause ;
47         défaut :
48             Pause ;
49     }
50 }
51
52 néant tk_CalculateVoltageCompensation () {
53     getBatteryVoltage (); // à partir de la bibliothéque Freenove_4WD_Car_for_Arduino.h
54     flotte voltageOffset = 7 - batteryVoltage; tk_VoltageCompensationToSpeed
55     = 30 * voltageOffset;
56 }
57
58 // lorsqu'une ligne noire sur un côté est détectée, la valeur du côté sera 0, ou la valeur est 1
59 u8 getTrackingSensorVal () {
60     u8 trackingSensorVal = 0;
61     trackingSensorVal = (digitalRead ( PIN_TRACKING_LEFT ) == 1? 1: 0) << 2 |
62     (digitalLire ( PIN_TRACKING_CENTER ) == 1? 1: 0) << 1 | (digitalLire ( PIN_TRACKING_RIGHT ) == 1? 1: 0) << 0;
63
64     revenir trackingSensorVal;
65 }
```

Exp1? Exp2: Exp3;

Si Exp1 est vrai, le résultat de ce code est Exp2. Si Exp1 est faux, le résultat de ce code est Exp3. Par exemple

Si y = 8;

var = (y <10)? 30: 40; alors var = 30 Si y = 10

var = (y <10)? 30: 40; alors var = 40

interrupteur ...

commutateur (var) {

Cas 1:

// fait quelque chose quand var est égal à 1

 Pause ;

Cas 2:

// fait quelque chose quand var vaut 2

 Pause ;

défaut :

// si rien d'autre ne correspond, faites la valeur par défaut // la valeur

par défaut est facultative

 Pause ;

}

Pour plus d'informations, veuillez consulter:

<https://www.arduino.cc/reference/en/language/structure/control-structure/switchcase/>

Chapitre 4 Contrôle IR

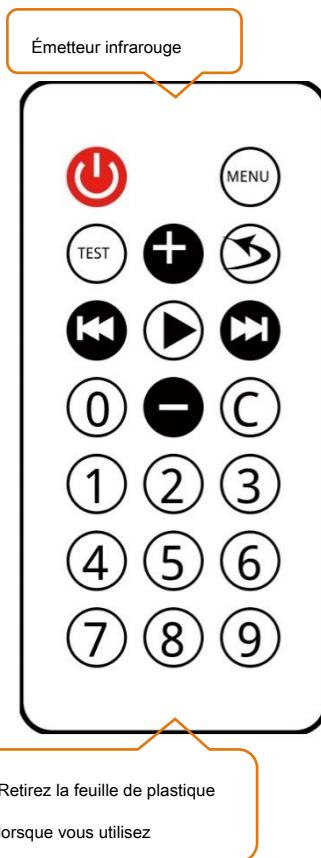
Dans ce chapitre, nous présenterons la télécommande et le récepteur infrarouges et fabriquerons une voiture télécommandée IR.

Si vous avez des préoccupations, n'hésitez pas à nous contacter via support@freenove.com

4.1 Télécommande et récepteur infrarouges

Télécommande infrarouge (IR) est un appareil qui fabriquera le tube d'émission infrarouge, avec différents codages. La télécommande infrarouge cont vous permet de changer de téléviseur lorsque vous vous en éloignez. La télécommande con Cette télécommande infrarouge utilise NEC fr

boutons érents
infrarouge avec
ng, etc. Ainsi,
tioning wheb



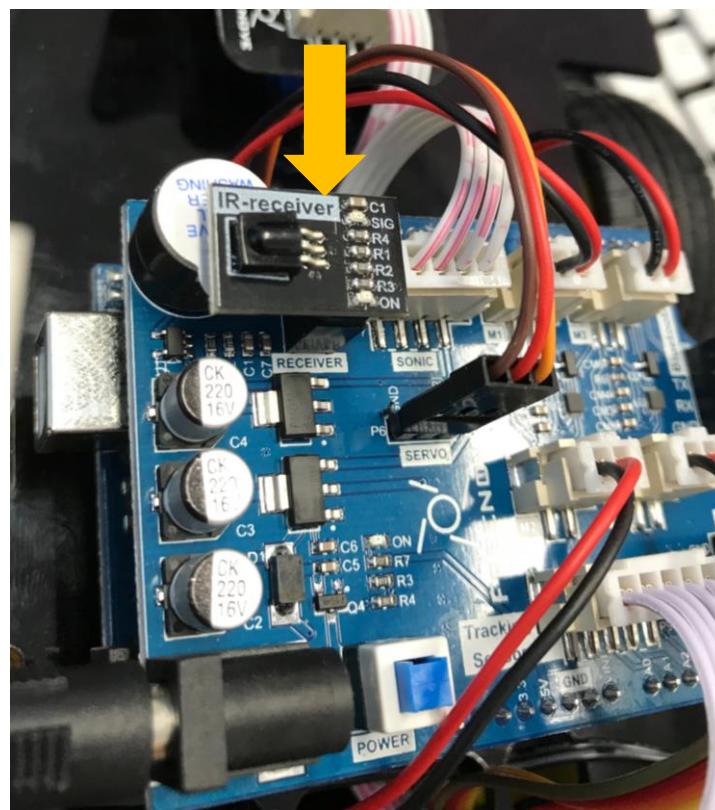
Récepteur infrarouge (IR) est un composant qui peut être réémis par la télécommande infrarouge. Broche DATA

lumière ared, donc nous ca
le reçu infrarouge

ect le signal



Installez le récepteur IR sur la voiture comme ci-dessous:



Code

04.1_IR_Receiver

Vous devez retirer le module Bluetooth M

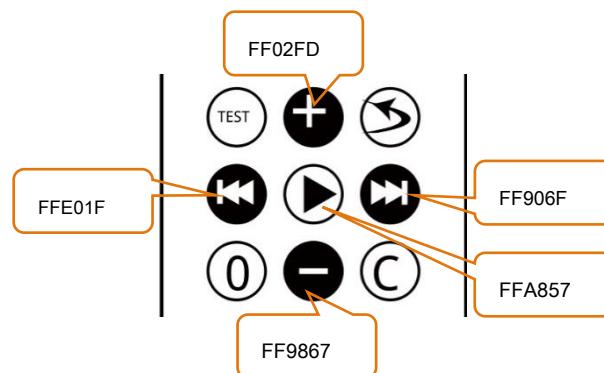
ou télécharger le code.

Ensuite, téléchargez le code dans Sketches \ 04.1_IR_Re Et puis

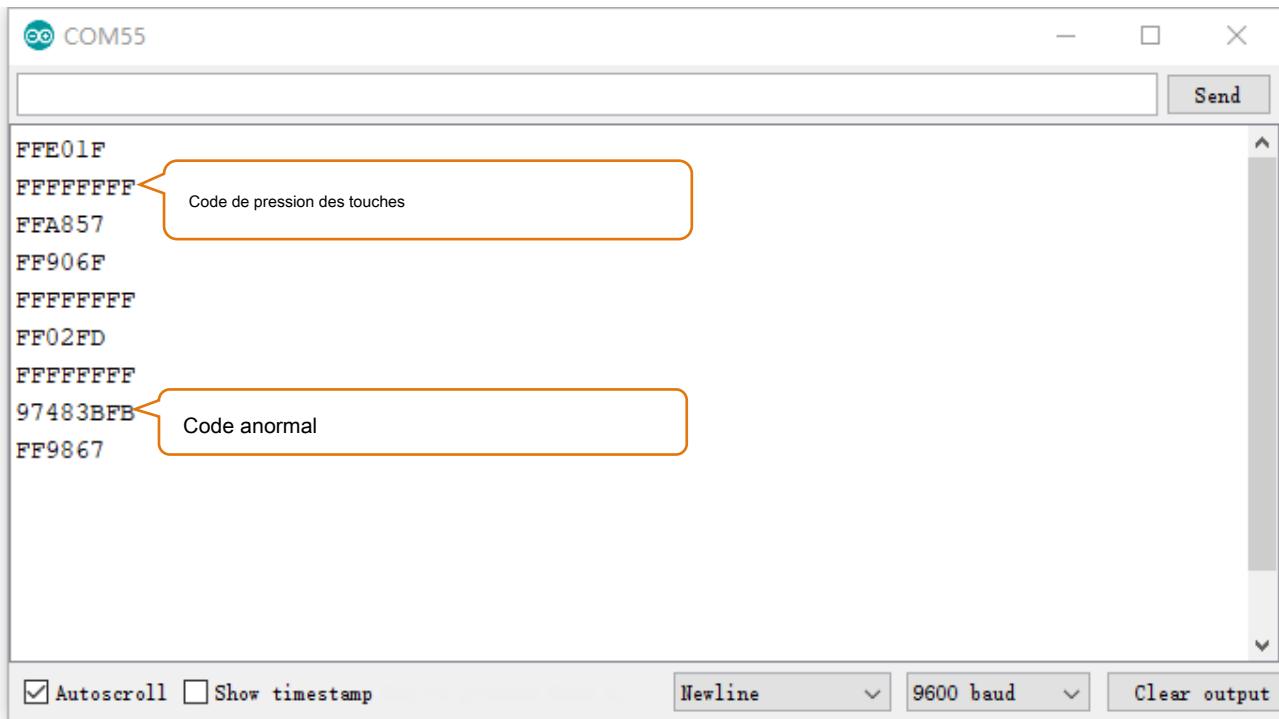
alimentation et ouvrez le moniteur série. nd appuyez sur les

retirer la pièce en plastique fermer

touches suivantes dans la télécommande IR.



Après avoir reçu le signal de l'émetteur infrarouge, le contrôleur reconnaîtra quelle touche a été enfoncée. Et imprimez-le sur le moniteur comme ci-dessous



Lorsque vous appuyez sur une touche pendant un certain temps, FFFFFFFF s'affiche. Il indique que la touche est en état de pression.

La communication infrarouge n'est pas très stable, elle est donc facilement interférée. Lorsqu'elle est utilisée, la LED émettrice de la télécommande doit être dirigée vers le récepteur. Sinon, il peut en résulter qu'aucune donnée ne sera reçue ou des données incorrectes seront reçues.

De plus, le code reçu est parfois anormal. À l'exception des codes de clé et FFFFFFFF, les autres codes reçus sont anormaux. Lorsque nous traitons le code, nous devons ignorer ce type de données.

Différer Le code clé ent correspond à une clé différente.

```

1 # comprendre <IRremote.h>           // inclure la bibliothèque IRremote // broche
2 # définir PIN_IRREMOTE_RECV 9        de réception infrarouge
3 IRrecv irrecv ( PIN_IRREMOTE_RECV ); // Crée un objet de classe utilisé pour recevoir
4 résultats decode_results;           // Créer un objet de classe de résultats de décodage
5
6 néant installer()
sept {
8     Serial.begin (9600);             // Initialisez le port série et réglez le débit en bauds sur 9600 // Démarrez le récepteur
9     irrecv.enableIRIn ();
dix }
11
12 néant boucle() {
13     si (irrecv.decode (& résultats)) { // En attente de décodage
14         Serial.println (results.value, HEX); // Imprimer les résultats décodés
15         irrecv.resume ();                  // Recevoir la valeur suivante
16     }
17     retard (100);
18 }
```

4.2 Voiture à distance IR

Nous utiliserons une télécommande infrarouge pour contrôler la voiture pour effectuer certaines actions.

Code

04.2_IR_Remote_Car

Vous devez d'abord supprimer le module de module Bluetooth lorsque vous téléchargez le code.

Tout d'abord, téléchargez le code dans Sketches \ 04.2_IR_Remote_Car et allumez la voiture. Graphique clé

	Définition de la clé	Code clé	Fonction
	IR_REMOTE_KEYCODE_UP	0xFF02FD	Avance
	IR_REMOTE_KEYCODE_DOWN	0xFF9867	reculer
	IR_REMOTE_KEYCODE_LEFT	0xFFE01F	Tournez à gauche
	IR_REMOTE_KEYCODE_RIGHT	0xFF906F	Tournez à droite
	IR_REMOTE_KEYCODE_CENTER 0xFFA857		Activer le buzzer

Le code est ci-dessous:

```

1 # comprendre <IRremote.h>
2 # comprendre "Freenove_4WD_Car_for_Arduino.h"
3
4 // définir la clé, le code ne peut pas être modifié.
5 # définir IR_REMOTE_KEYCODE_UP          0xFF02FD
6 # définir IR_REMOTE_KEYCODE_DOWN        0xFF9867
7 # définir IR_REMOTE_KEYCODE_LEFT        0xFFE01F
8 # définir IR_REMOTE_KEYCODE_RIGHT       0xFF906F
9 # définir IR_REMOTE_KEYCODE_CENTER     0xFFA857
10
11 # définir IR_UPDATE_TIMEOUT           120
12 # définir IR_CAR_SPEED                180
13
14 IRrecv irrecv ( PIN_IRREMOTE_RECV );
15 résultats decode_results;
16 u32 currentKeyCode, lastKeyCode;
17 boolen isStopFromIR = faux ; u32
18 lastIRUpdateTime = 0;
19
20 nant installer() {

```

```

21     irrecv.enableIRIn(); // Démarrer le récepteur
22 }
23
24 néant boucle() {
25     si (irrecv.decode (& résultats)) {
26         isStopFromIR = faux ;
27         currentKeyCode = results.value;
28         si (currentKeyCode! = 0xFFFFFFFF) {
29             lastKeyCode = currentKeyCode;
30         }
31         commutateur (lastKeyCode) {
32             Cas IR_REMOTE_KEYCODE_UP :
33                 motorRun ( IR_CAR_SPEED , IR_CAR_SPEED ); //Avance
34                 Pause ;
35             Cas IR_REMOTE_KEYCODE_DOWN :
36                 motorRun (- IR_CAR_SPEED , - IR_CAR_SPEED ); //reculer
37                 Pause ;
38             Cas IR_REMOTE_KEYCODE_LEFT :
39                 motorRun (- IR_CAR_SPEED , IR_CAR_SPEED );           //Tourner à gauche
40                 Pause ;
41             Cas IR_REMOTE_KEYCODE_RIGHT :
42                 motorRun ( IR_CAR_SPEED , - IR_CAR_SPEED );        //Tournez à droite
43                 Pause ;
44             Cas IR_REMOTE_KEYCODE_CENTER :
45                 setBuzzer ( vrai );                                // activer le buzzer
46                 Pause ;
47             défaut :
48                 Pause ;
49         }
50         irrecv.resume (); // Recevoir la valeur suivante
51         lastIRUpdateTime = millis (); // noter l'heure actuelle
52     }
53     autre {
54         si (millis () - lastIRUpdateTime> IR_UPDATE_TIMEOUT ) {
55             si (! isStopFromIR) {
56                 isStopFromIR = vrai ;
57                 motorRun (0, 0);
58                 setBuzzer ( faux );
59             }
60             lastIRUpdateTime = millis ();
61         }
62     }
63 }
64

```

Etant donné que la période d'émission de la télécommande infrarouge est de 108 ms, le temps de temporisation du signal est réglé à une valeur approximative de 120 ms. Si le délai d'attente est inférieur à 108 ms, les données seront lues à l'avance et ne pourront pas être lues. Cela entraînera un malentendu selon lequel le signal est perdu. Si le délai d'expiration est trop grand, par exemple 1000 ms, le retard de l'action de la voiture sera plus grave.

```
11 # définir IR_UPDATE_TIMEOUT      120
.....  
54     si (millis () - lastIRUpdateTime > IR_UPDATE_TIMEOUT) {  
55         si (! isStopFromIR) {  
56             isStopFromIR = vrai ;  
57             motorRun (0, 0);  
58             setBuzzer ( faux );  
59         }  
60         lastIRUpdateTime = millis ();  
61     }  
.....
```

Vous pouvez modifier la valeur de **IR_CAR_SPEED** pour changer la vitesse de la voiture dans la plage (0 ~ 255). En fait, puisque le besoin moteur en tension gh pour faire bouger la voiture de sorte que la valeur minimale ne soit pas 0.

```
12 # définir IR_CAR_SPEED      180
```

millis ()

Renvoie le nombre de millisecondes écoulées depuis que la carte Arduino a commencé à exécuter le **actuel** programme

lastIRUpdateTime = millis (); // noter l'heure actuelle

}

autre {

si (millis () - lastIRUpdateTime)

millis () - lastIRUpdateTime est le temps entre lastIRUpdateTime et la ligne actuelle.

Pour plus de détails, veuillez consulter:

<https://www.arduino.cc/reference/en/language/functions/time/millis/>

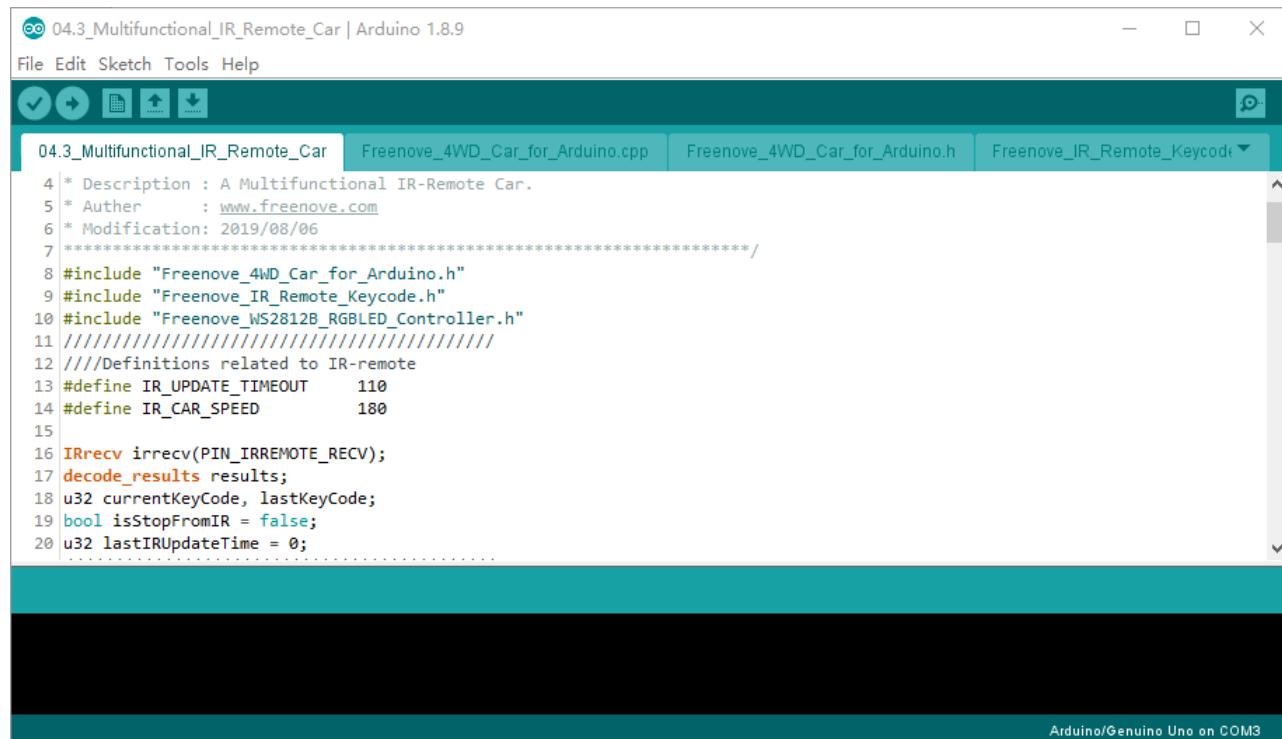
4.3 Voiture à distance infrarouge multifonctionnelle

Dans ce projet, nous ajouterons une fonction de barre LED par rapport à la dernière section, de sorte que la voiture télécommandée infrarouge aura plus de fonctions.

Télécharger le code et exécuter

Vous devez d'abord supprimer le module de module Bluetooth lorsque vous téléchargez le code.

Ouvrez le code dans **Croquis \ 04.4_One_Code_Multifunctional_IR_Remote_Car**. Téléchargez-le dans la voiture. Vous pouvez également choisir Sketches \ 04.3_Multifunctional_IR_Remote_Car



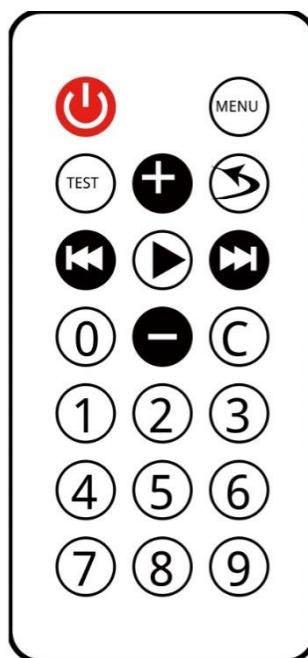
The screenshot shows the Arduino IDE interface with the sketch titled "04.3_Multifunctional_IR_Remote_Car" open. The code is as follows:

```
04.3_Multifunctional_IR_Remote_Car | Arduino 1.8.9
File Edit Sketch Tools Help
04.3_Multifunctional_IR_Remote_Car Freenove_4WD_Car_for_Arduino.cpp Freenove_4WD_Car_for_Arduino.h Freenove_IR_Remote_Keycode.h
4 * Description : A Multifunctional IR-Remote Car.
5 * Author : www.freenove.com
6 * Modification: 2019/08/06
7 ****
8 #include "Freenove_4WD_Car_for_Arduino.h"
9 #include "Freenove_IR_Remote_Keycode.h"
10 #include "Freenove_WS2812B_RGBLED_Controller.h"
11 ///////////////////////////////////////////////////
12 //Definitions related to IR-remote
13 #define IR_UPDATE_TIMEOUT 110
14 #define IR_CAR_SPEED 180
15
16 IRrecv irrecv(PIN_IRREMOTE_RECV);
17 decode_results results;
18 u32 currentKeyCode, lastKeyCode;
19 bool isStopFromIR = false;
20 u32 lastIRUpdateTime = 0;
```

Arduino/Genuino Uno on COM3

Une fois le code téléchargé, vous contrôlez la voiture et d'autres fonctions. La table c:

le contrôle de
e suivant



Graphique clé	Définition de la clé	Code clé	Fonction
	IR_REMOTE_KEYCODE_UP	0xFF02FD	avancer
	IR_REMOTE_KEYCODE_DOWN	0xFF9867	reculer
	IR_REMOTE_KEYCODE_LEFT	0xFFE01F	Tournez à gauche
	IR_REMOTE_KEYCODE_RIGHT	0xFF906F	Tournez à droite
	IR_REMOTE_KEYCODE_CENTER	0xFFA857	Activer le buzzer
	IR_REMOTE_KEYCODE_1	0xFF30CF	Faites fonctionner la LED en mode 1 pour faire défiler la couleur de l'arc-en-ciel.
	IR_REMOTE_KEYCODE_4	0xFF10EF	Faire fonctionner la LED en mode 2, en changeant la couleur de la LED d'eau
	IR_REMOTE_KEYCODE_2	0xFF18E7	La couleur de la barre LED change plus rapidement. La couleur provient de ColorWheel.
	IR_REMOTE_KEYCODE_3	0xFF7A85	La couleur de la barre LED change plus lentement.
	IR_REMOTE_KEYCODE_5	0xFF38C7	La période de cycle de la barre LED est diminuée et la barre de LED change à une vitesse plus rapide
	IR_REMOTE_KEYCODE_6	0xFF5AA5	La période de cycle de la barre LED est augmentée et la barre de LED change à une vitesse plus lente

Code

Vous devez d'abord supprimer le module de module Bluetooth lorsque vous téléchargez le code.

Ce projet a de nombreux labels.

«Freenove_IR_Remote_Keycode.h» est utilisé pour enregistrer le code clé de la télécommande IR.

Freenove_IR_Remote_Keycode.h

```
1 // Freenove_IR_Remote_Keycode.h
2 #ifndef _FREENOVE_IR_REMOTE_KEYCODE_H
3 # définir _FREENOVE_IR_REMOTE_KEYCODE_H
4 # si défini ( ARDUINO ) && ARDUINO > = 100
5     # comprendre "arduino.h"
6     # autre
7     # comprendre "WProgram.h"
8 # fin si
9 # comprendre <IRremote.h>
10
11 # définir IR_REMOTE_KEYCODE_POWER          0xFFA25D
12 # définir IR_REMOTE_KEYCODE_MENU           0xFF629D
13 # définir IR_REMOTE_KEYCODE_MUTE           0FFE21D
14
15 # définir IR_REMOTE_KEYCODE_MODE          0xFF22DD
16 # définir IR_REMOTE_KEYCODE_UP            0xFF02FD
17 # définir IR_REMOTE_KEYCODE_BACK          0FFC23D
18
19 # définir IR_REMOTE_KEYCODE_LEFT          0FFE01F
20 # définir IR_REMOTE_KEYCODE_CENTER        0FFA857
21 # définir IR_REMOTE_KEYCODE_RIGHT         0FF906F
22
23 # définir IR_REMOTE_KEYCODE_0             0FF6897
24 # définir IR_REMOTE_KEYCODE_DOWN          0FF9867
25 # définir IR_REMOTE_KEYCODE_OK            0FFB04F
26
27 # définir IR_REMOTE_KEYCODE_1             0FF30CF
28 # définir IR_REMOTE_KEYCODE_2             0FF18E7
29 # définir IR_REMOTE_KEYCODE_3             0FF7A85
30
31 # définir IR_REMOTE_KEYCODE_4             0FF10EF
32 # définir IR_REMOTE_KEYCODE_5             0FF38C7
33 # définir IR_REMOTE_KEYCODE_6             0FF5AA5
34
35 # définir IR_REMOTE_KEYCODE_7             0FF42BD
36 # définir IR_REMOTE_KEYCODE_8             0FF4AB5
37 # définir IR_REMOTE_KEYCODE_9             0FF52AD
```

38	# fin si
----	----------

04.3_Multifunctional_IR_Remote_Car.ino

```
1      # comprendre "Freenove_4WD_Car_for_Arduino.h"
2      # comprendre "Freenove_IR_Remote_Keycode.h"
3      # comprendre "Freenove_WS2812B_RGBLED_Controller.h"
4      ///////////////////////////////////////////////////////////////////
5      /// Définitions relatives à la télécommande IR
6      # définir IR_UPDATE_TIMEOUT          110
7      # définir IR_CAR_SPEED             180
8
9      IRrecv irrecv ( PIN_IRREMOTE_RECV );
10     decode_results résultats ;
11     u32 currentKeyCode , lastKeyCode ;
12     boolen isStopFromIR = faux ;
13     u32 lastIRUpdateTime = 0;
14     ///////////////////////////////////////////////////////////////////
15     /// Définitions relatives à la bande LED
16     # définir STRIP_I2C_ADDRESS 0x20
17     # définir STRIP_LEDS_COUNT      dix
18
19     u8 colorPos = 0;
20     u8 colorStep = 50;
21     u8 stripDisplayMode = 1;
22     u8 currentLedIndex = 0;
23     u16 stripDisplayDelay = 100;
24     u32 lastStripUpdateTime = 0;
25     Freenove_WS2812B_Controller bande ( STRIP_I2C_ADDRESS , STRIP_LEDS_COUNT , TYPE_GRB );
26     ///////////////////////////////////////////////////////////////////
27
28     néant installer () {
29         bande . commencer ();
30         irrecv . enableIRIn (); // Démarrer le récepteur
31     }
32
33     néant boucle () {
34         si ( irrecv . décoder (& résultats )) {
35             isStopFromIR = faux ;
36             currentKeyCode = résultats . valeur ;
37             si ( currentKeyCode != 0xFFFFFFFF ) {
38                 lastKeyCode = currentKeyCode ;
39             }
40             commutateur ( lastKeyCode ) {
41                 Cas IR_REMOTE_KEYCODE_UP :
42                     motorRun ( IR_CAR_SPEED , IR_CAR_SPEED );
```

```
43         Pause ;  
44  
45     Cas IR_REMOTE_KEYCODE_DOWN :  
46         motorRun (- IR_CAR_SPEED , - IR_CAR_SPEED );  
47         Pause ;  
48  
49     Cas IR_REMOTE_KEYCODE_LEFT :  
50         motorRun (- IR_CAR_SPEED , IR_CAR_SPEED );  
51         Pause ;  
52  
53     Cas IR_REMOTE_KEYCODE_RIGHT :  
54         motorRun ( IR_CAR_SPEED , - IR_CAR_SPEED );  
55         Pause ;  
56  
57     Cas IR_REMOTE_KEYCODE_CENTER :  
58         setBuzzer ( vrai );  
59         Pause ;  
60  
61     Cas IR_REMOTE_KEYCODE_0 :  
62         Pause ;  
63  
64     Cas IR_REMOTE_KEYCODE_1 :  
65         stripDisplayMode = 1;  
66         Pause ;  
67  
68     Cas IR_REMOTE_KEYCODE_2 :  
69         colorStep += 5;  
70         si ( colorStep > 100) {  
71  
72             colorStep = 100;  
73         }  
74         Pause ;  
75  
76     Cas IR_REMOTE_KEYCODE_3 :  
77         colorStep -= 5;  
78         si ( colorStep <5) {  
79  
80             colorStep = 5;  
81         }  
82         Pause ;  
83  
84     Cas IR_REMOTE_KEYCODE_4 :  
85         stripDisplayMode = 0;  
86         Pause ;  
87  
88     Cas IR_REMOTE_KEYCODE_5 :  
89         stripDisplayDelay -= 20;  
90         si ( stripDisplayDelay <20) {  
91  
92             stripDisplayDelay = 20;  
93         }  
94         Pause ;  
95  
96     Cas IR_REMOTE_KEYCODE_6 :  
97         stripDisplayDelay += 20;
```

```
87         si ( stripDisplayDelay > 300 ) {
88
89             stripDisplayDelay = 300;
90         }
91         Pause ;
92     }
93     irrecv . CV (); // Recevoir la valeur suivante
94     lastIRUpdateTime = millis ();
95 }
96 autre {
97     si ( millis () - lastIRUpdateTime > IR_UPDATE_TIMEOUT ) {
98
99         si ( ! isStopFromIR ) {
100             isStopFromIR = vrai ;
101             motorRun ( 0, 0 );
102             setBuzzer ( faux );
103         }
104         lastIRUpdateTime = millis ();
105     }
106 }
107 commutateur ( stripDisplayMode )
108 {
109 Cas 0:
110     si ( millis () - lastStripUpdateTime > stripDisplayDelay ) {
111
112         pour ( int je = 0; je < STRIP_LEDS_COUNT ; je ++ ) {
113             bande . setLedColorData ( je , bande . Roue ( colorPos + je * 25));
114         }
115         bande . montrer ();
116         colorPos += colorStep ;
117         lastStripUpdateTime = millis ();
118     }
119     Pause ;
120 Cas 1:
121     si ( millis () - lastStripUpdateTime > stripDisplayDelay ) {
122
123         bande . setLedColor ( currentLedIndex , bande . Roue ( colorPos ));
124         currentLedIndex++;
125         si ( currentLedIndex == STRIP_LEDS_COUNT ) {
126
127             currentLedIndex = 0;
128             colorPos += colorStep ;
129         }
130         lastStripUpdateTime = millis ();
```

```
131      }
132      Pause ;
133  défaut :
134      Pause ;
135  }
136 }
```

Par rapport au projet précédent, ce code a plus de branches de boîtier de commutation du code de clé à distance pour faire rouler la barre lumineuse.

```
1 .....  
2 Cas IR_REMOTE_KEYCODE_1 :  
3     stripDisplayMode = 1;  
4     Pause ;  
5 Cas IR_REMOTE_KEYCODE_2 :  
6     colorStep += 5;  
7     si ( colorStep > 100) {  
8         colorStep = 100;  
9     }  
10    Pause ;  
11 Cas IR_REMOTE_KEYCODE_3 :  
12     colorStep -= 5;  
13     si ( colorStep < 5) {  
14         colorStep = 5;  
15     }  
16     Pause ;  
17 Cas IR_REMOTE_KEYCODE_4 :  
18     stripDisplayMode = 0;  
19     Pause ;  
20 Cas IR_REMOTE_KEYCODE_5 :  
21     stripDisplayDelay -= 20;  
22     si ( stripDisplayDelay < 20) {  
23         stripDisplayDelay = 20;  
24     }  
25     Pause ;  
26 Cas IR_REMOTE_KEYCODE_6 :  
27     stripDisplayDelay += 20;  
28     si ( stripDisplayDelay > 300) {  
29         stripDisplayDelay = 300;  
30     }  
31     Pause ;
```

.....

Dans la boucle (), contrôlez le comportement de la barre lumineuse en fonction des variables ci-dessus. La variable **stripDisplayMode** est utilisé pour contrôler le mode d'affichage de la LED. La variable **stripDisplayDelay** est utilisé pour contrôler le temps de cycle de la LED, c'est-à-dire la vitesse de changement. La variable **colorStep** est utilisé pour contrôler le changement de valeur de couleur pour chaque itération.

```
1  commutateur ( stripDisplayMode )
2  {
3      Cas 0:
4          si ( millis () - lastStripUpdateTime > stripDisplayDelay ) {
5
6              pour ( int je = 0; je < STRIP_LEDS_COUNT ; je ++ ) {
7                  bande . setLedColorData ( je , bande . Roue ( colorPos + je * 25));
8
9                  bande . montrer ();
10                 colorPos += colorStep ;
11                 lastStripUpdateTime = millis ();
12             }
13             Pause ;
14         Cas 1:
15             si ( millis () - lastStripUpdateTime > stripDisplayDelay ) {
16
17                 bande . setLedColor ( currentLedIndex , bande . Roue ( colorPos ));
18                 currentLedIndex++;
19                 si ( currentLedIndex == STRIP_LEDS_COUNT ) {
20
21                     currentLedIndex = 0;
22                     colorPos += colorStep ;
23                 }
24                 lastStripUpdateTime = millis ();
25             }
26             Pause ;
27         défaut :
28             Pause ;
29     }
```

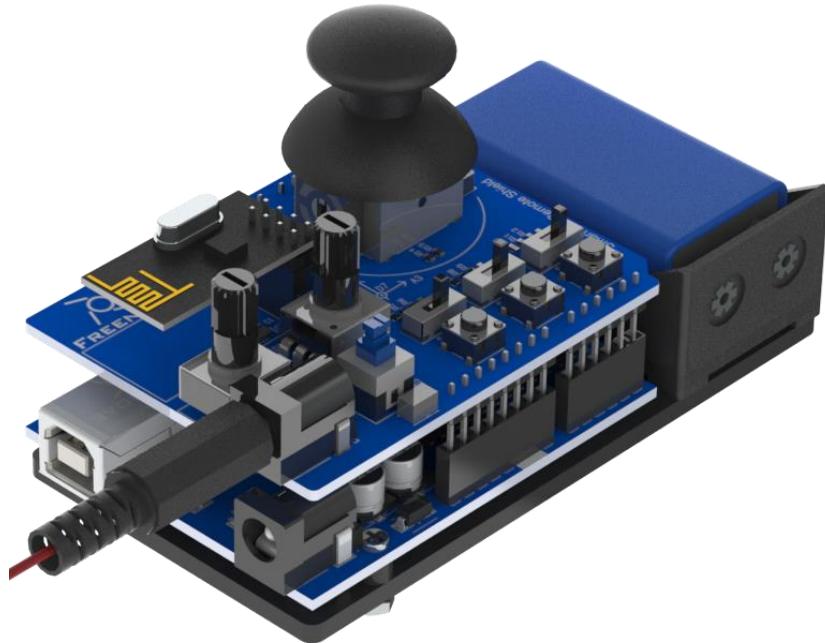
Chapitre 5 Télécommande RF

Si vous avez acheté le kit sans télécommande RF, vous pouvez ignorer ce chapitre. Si vous avez des préoccupations, n'hésitez pas à nous contacter via support@freenove.com

5.1 Télécommande

Téléchargez le tutoriel et le code pour **assembler** télécommande.

https://github.com/Freenove/Freenove_Remote_Control_Kit/raw/master/Tutorial.pdf



Pour toutes les ressources, veuillez consulter https://github.com/Freenove/Freenove_Remote_Control_Kit

Télécharger le code et exécuter

Connectez la télécommande à l'ordinateur. Et téléchargez le code dans Sketches \ 05.1_RF24_Remote_Controller. Mettez ensuite la télécommande RF sous tension. **Vous devez d'abord supprimer BluetoothModule lorsque vous téléchargez le code.**

```

05.1_RF24_Remote_Controller | Arduino 1.8.9
File Edit Sketch Tools Help
05.1_RF24_Remote_Controller
1 // ****
2 * Filename      : RF24_Remote_Controller.ino
3 * Product       : Freenove 4WD Car for UNO
4 * Description   : Project 05.1 - Code for RF24 Remote Controller.
5 * Author        : www.freenove.com
6 * Modification: 2019/08/06
7 ****
8 // NRF24L01
9 #include <SPI.h>
10 #include "RF24.h"
11 RF24 radio(9, 10);           // define the object to control NRF24L01
12 const byte addresses[6] = "Free1"; // define communication address which should correspond to remote co
13 // wireless communication
14 int dataWrite[8];           // define array used to save the write data
< >
4
Arduino/Genuino Uno on COM55

```

05.1_RF24_Remote_Controller

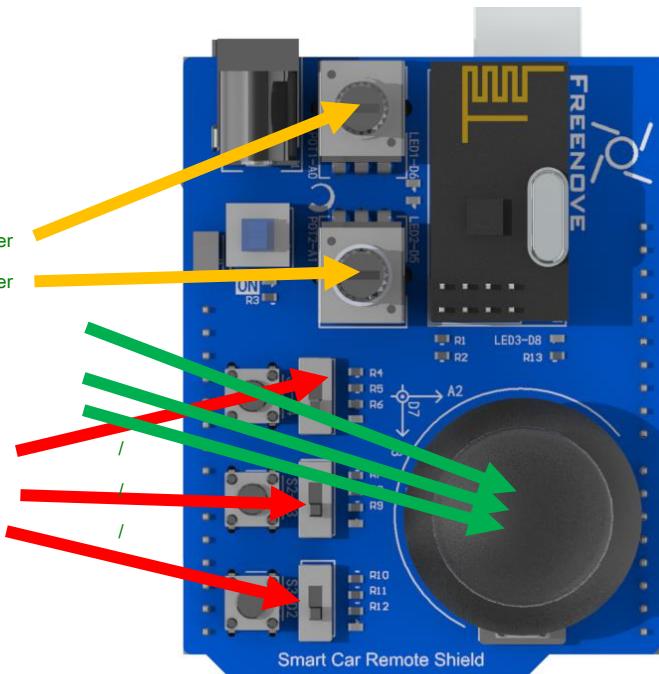
Le code est ci-dessous:

1	// RF24L01
2	# comprendre <SPI.h>
3	# comprendre "RF24.h"
4	Radio RF24 (9, 10); // définir l'objet à contrôler RF24L01
5	const adresses d'octet [6] = "Free1"; // définir l'adresse de communication qui doit correspondre au tableau de commande de la voiture
6	
sept	// Communication sans fil
8	int dataWrite [8]; // définir le tableau utilisé pour sauvegarder les données d'écriture
9	// broche
dix	const int pot1Pin = A0, // définir le potentiomètre POT1 // définir
11	pot2Pin = A1, le potentiomètre POT2
12	joystickXPin = A2, // définir la broche pour la direction X du joystick // définir la
13	joystickYPin = A3, broche pour la direction Y du joystick // définir la broche pour
14	joystickZPin = 7, la direction Z du joystick // définir la broche pour S1
15	s1Pin = 4, // définir la broche pour S2 //
16	s2Pin = 3, définir la broche pour S3
17	s3Pin = 2,

```

18      led1Pin = 6;           // définir la broche pour LED1 qui est proche de POT1 et utilisée pour
19      indiquer l'état de POT1
20      led2Pin = 5;           // définir la broche pour LED2 qui est proche de POT2 et utilisée pour
21      indiquer l'état du POT2
22      led3Pin = 8;           // définir la broche pour LED3 qui est proche de RF24L01 et utilisée pour
23      indiquer l'état de RF24L01
24
25  néant installer() {
26      // RF24L01
27      radio.begin();          // initialiser RF24
28      radio.setPAlevel (RF24_PA_MAX); // régler le niveau de l'amplificateur de puissance (PA) //
29      radio.setDataRate (RF24_1MBPS); // régler le débit de données dans l'air
30      radio.setRetries (0, 15); // définit le nombre et le délai des tentatives // ouvre un
31      radio.openWritingPipe (adresses); // tube pour l'écriture
32      radio.openReadingPipe (1, adresses); // ouvre un tube pour lire
33      radio.stopListening(); // arrêter d'écouter les messages entrants
34
35  // broche
36  pinMode (joystickZPin, INPUT); // définit led1Pin en mode d'entrée // définit
37  pinMode (s1Pin, INPUT);        s1Pin en mode d'entrée // définit s2Pin en
38  pinMode (s2Pin, INPUT);        mode d'entrée // définit s3Pin en mode
39  pinMode (s2Pin, INPUT);        d'entrée
40  pinMode (led1Pin, OUTPUT);    // définir led1Pin en mode sortie // définir
41  pinMode (led2Pin, OUTPUT);    led2Pin en mode sortie // définir led3Pin en
42  pinMode (led3Pin, SORTIE);   mode sortie
43 }
44
45  néant boucle()
46 {
47      // met les valeurs de rocker, switch et pote
48      dataWrite [0] = analogRead (pot1Pin); // enregistrer
49      dataWrite [1] = analogRead (pot2Pin); // enregistrer
50      dataWrite [2] = analogRead (joystickXPin); /
51      dataWrite [3] = analogRead (joystickYPin); /
52      dataWrite [4] = digitalRead (joystickZPin); /
53      dataWrite [5] = digitalRead (s1Pin);
54      dataWrite [6] = digitalRead (s2Pin);
55      dataWrite [7] = digitalRead (s3Pin);
56
57      // écrire des données radio
58      si (radio.write (dataWrite, taille de (dataWrite
59      {
60          digitalWrite (led3Pin, HIGH);
61          retard (20);

```



```

62   digitalWrite (led3Pin, LOW);
63 }
64
65 // faire en sorte que la LED émette une luminosité différente de la lumière selon l'analogique du potentiomètre
66 analogWrite (led1Pin, carte (dataWrite [0], 0, 1023, 0, 255)); analogWrite (led2Pin,
67 map (dataWrite [1], 0, 1023, 0, 255));
68 }
```

5	const adresses d'octet [6] = "Free1";
---	--

Vous pouvez changer l'adresse si vous avez deux télécommandes Freenove pour contrôler différentes voitures, comme Free2. Mais il est

necessary pour le garder identique à l'adresse dans la voiture.

```

48   dataWrite [0] = analogRead (pot1Pin);           // (0 ~ 1023) sauvegarde les données du potentiomètre 1 // (0 ~ 1023)
49   dataWrite [1] = analogRead (pot2Pin);           sauvegarde les données du potentiomètre 2
50   dataWrite [2] = analogRead (joystickXPin); // (0 ~ 1023) sauvegarde les données de la direction X du joystick
51   dataWrite [3] = analogRead (joystickYPin); // (0 ~ 1023) sauvegarde les données de la direction Y du joystick
52   dataWrite [4] = digitalRead (joystickZPin); // (0,1) sauvegarde les données de la direction Z du joystick
53   dataWrite [5] = digitalRead (s1Pin);           // (0,1) sauvegarde les données du commutateur 1
54   dataWrite [6] = digitalRead (s2Pin);           // (0,1) sauvegarde les données du commutateur 2
55   dataWrite [7] = digitalRead (s3Pin);           // (0,1) sauvegarde les données du commutateur 3
```

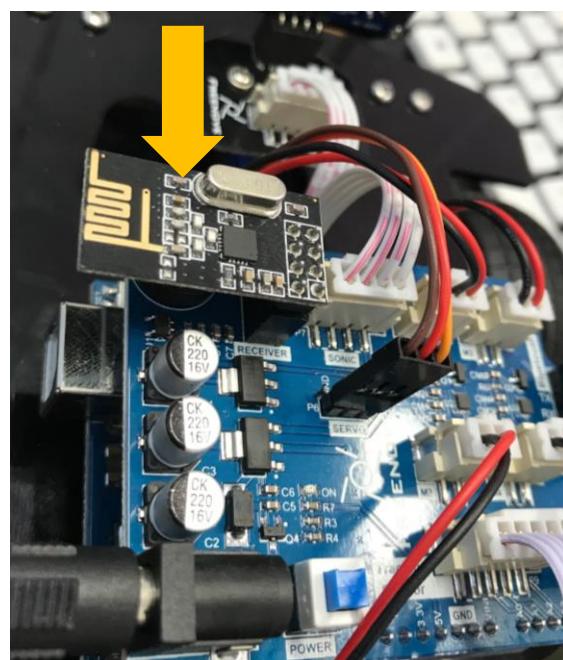
Lorsque le joystick est au centre, sa valeur est d'environ x = 512, y = 512 Lorsque le joystick

est enfoncé, z = 0; lorsqu'il n'est pas enfoncé, z = 1.

Les commutateurs sont les mêmes, lorsque le commutateur est activé, s = 0; lorsque l'interrupteur est désactivé, s = 1.

5.2 Recevoir les données de la télécommande RF

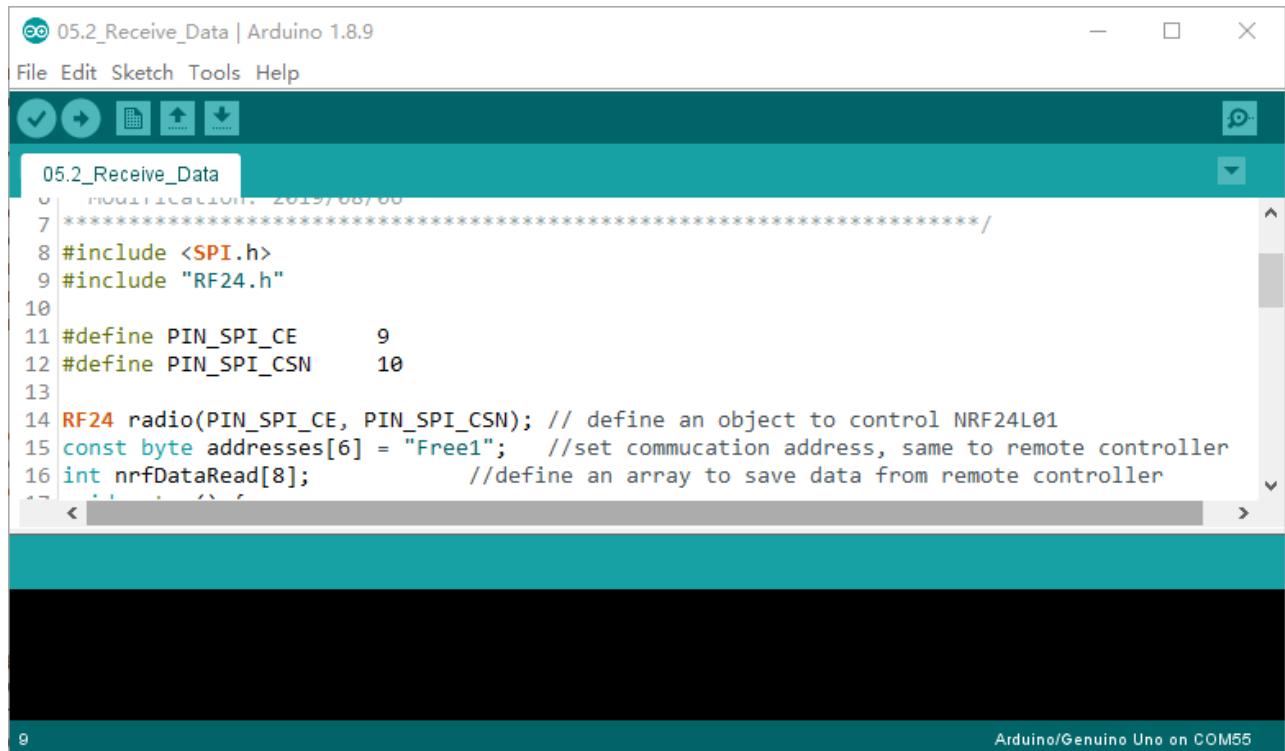
Installez le module RF sur la voiture. Le module RF et le récepteur IR utilisent les mêmes broches de récepteur.



Télécharger le code et exécuter

Ensuite, connectez la carte de commande de la voiture à l'ordinateur. Et téléchargez le code dans Sketches \ 05.2_Receive_Data.

Vous devez d'abord supprimer le module Bluetooth lorsque vous téléchargez le code.



The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** 05.2_Receive_Data | Arduino 1.8.9
- Menu Bar:** File Edit Sketch Tools Help
- Toolbar:** Includes icons for Save, Run, Upload, and Download.
- Sketch Area:** Displays the code for "05.2_Receive_Data".

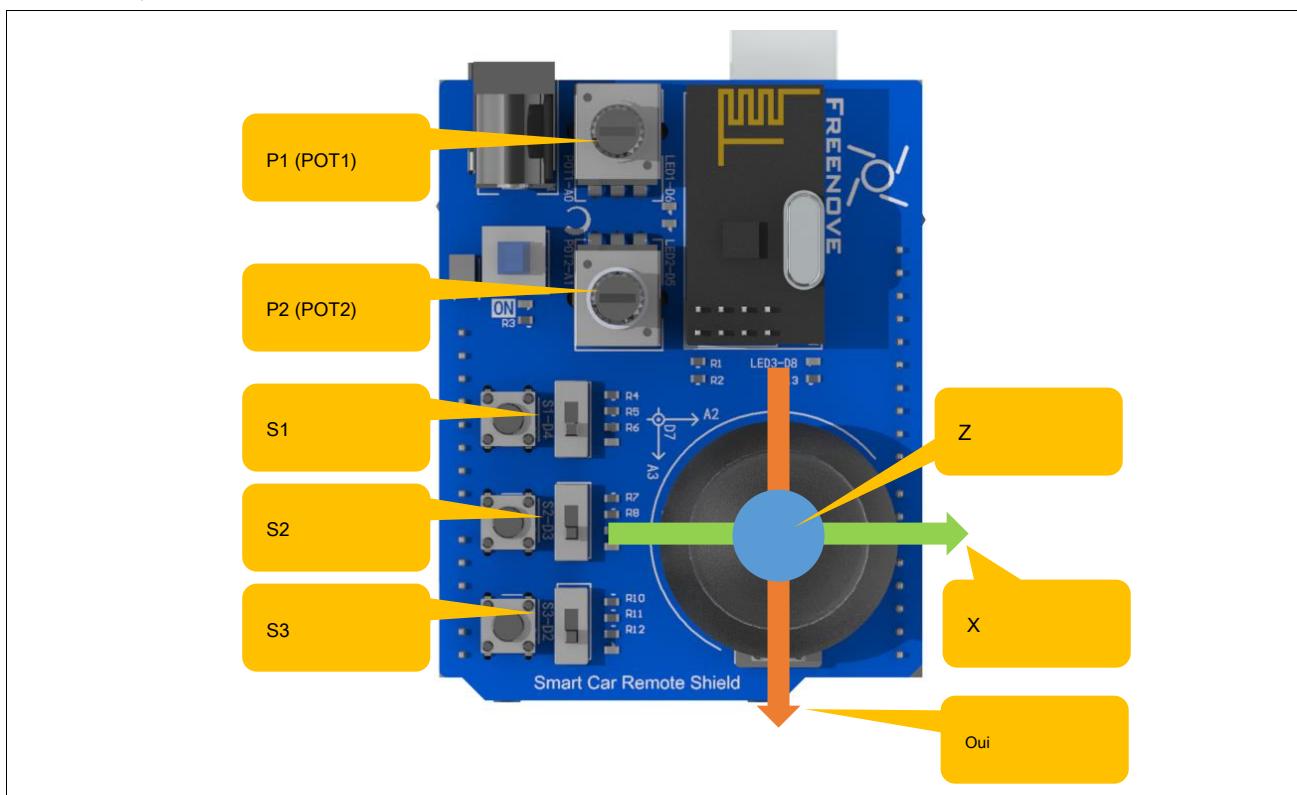
```
05.2_Receive_Data
v |   2019-08-06 14:14:11
7 ****
8 #include <SPI.h>
9 #include "RF24.h"
10
11 #define PIN_SPI_CE      9
12 #define PIN_SPI_CSN     10
13
14 RF24 radio(PIN_SPI_CE, PIN_SPI_CSN); // define an object to control NRF24L01
15 const byte addresses[6] = "Free1";    //set communication address, same to remote controller
16 int nrfDataRead[8];                //define an array to save data from remote controller
17 . . .
< . . . >
```
- Status Bar:** Arduino/Genuino Uno on COM55

Alimentez la télécommande.

Gardez la carte de commande de la voiture connectée à l'ordinateur via un câble USB et ouvrez le moniteur série. Vous verrez le contenu comme ci-dessous.

Utilisez la télécommande et observez le changement de données.

Données aux composants



Code

05.2_Receive_Data

Le code est ci-dessous.

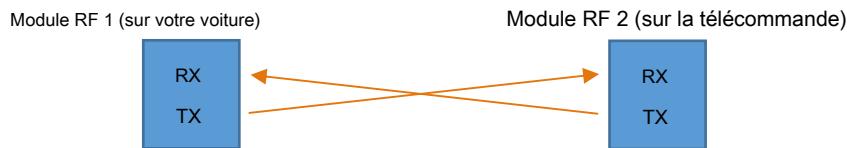
```

1   # comprendre <SPI.h>
2   # comprendre "RF24.h"
3
4   # définir PIN_SPI_CE          9
5   # définir PIN_SPI_CSN        dix
6
7
8   Radio RF24 ( PIN_SPI_CE , PIN_SPI_CSN ); // définir un objet pour contrôler RF24L01
9   const adresses d'octet [6] = "Free1";           // définir l'adresse de commutation, identique à la télécommande
10  int RFDataRead [8];                           // définir un tableau pour enregistrer les données de la télécommande
11
12  néant installer() {
13      Serial.begin (9600);
14
15  // Etapes d'initialisation RF24L01
16
17  si (radio.begin ()) {                         // initialiser RF24
18      radio.setPAlevel (RF24_PA_MAX);           // régler le niveau de l'amplificateur de puissance (PA) //
19      radio.setDataRate (RF24_1MBPS);            // régler le débit de données dans l'air
20      radio.setRetries (0, 15);                 // définit le nombre et le délai des tentatives // ouvre un
21      radio.openWritingPipe (adresses);          // tube pour l'écriture
22      radio.openReadingPipe (1, adresses);         // ouvre un tube pour lire
23      radio.startListening ();                  // démarrer la surveillance, démarrer l'écoute sur les tubes ouverts
24      Serial.println ( "Commencez à écouter les données distantes ..." );
25
26  }
27
28  autre {
29      Serial.println ( "Pas trouvé la puce RF!" );
30  }
31
32
33  néant boucle() {
34      si (radio disponible ()) {                // si recevoir les données // lire
35          tandis que (radio disponible ()) {      toutes les données
36              radio.read (RFDataRead, taille de (RFDataRead));    // lire les données
37          }
38          Serial.print ( «P1 / P2 / X / Y / Z / S1 / S2 / S3:» );
39          pour ( int i = 0; je < taille de (RFDataRead) / 2; i ++ ) {
40              Serial.print (RFDataRead [i]);
41              Serial.print ( '\t' );
42          }
43          Serial.print ( '\n' );
44      }
45  }

```

```
40 }  
41 }
```

La communication est la suivante:



```
radio.write (dataWrite, taille de (dataWrite) Transférez les données.
```

Tous les deux des codes pour voiture et pour télécommande ont le contenu suivant:

```
5 const adresses d'octet [6] = "Free1" ;  
.....  
18 radio.openWritingPipe (adresses); // ouvre un tube pour écrire  
19 radio.openReadingPipe (1, adresses); // ouvre un tube pour lire
```

Ceci est utilisé pour définir les adresses d'écriture (émission) et de lecture (réception) du module RF24L01. Le module B peut recevoir les données du module A, uniquement lorsque l'adresse d'écriture du module A est la même que l'adresse de lecture du module B. Les propres adresses d'écriture et de lecture du module peuvent être identiques ou différentes. Lorsqu'il y a de nombreux modules RF à proximité, les adresses d'écriture et de lecture peuvent être modifiées pour éviter les interférences d'autres appareils.

Après avoir reçu les données de la télécommande, nous reconnaîtrons quel composant est utilisé en fonction du changement de données.

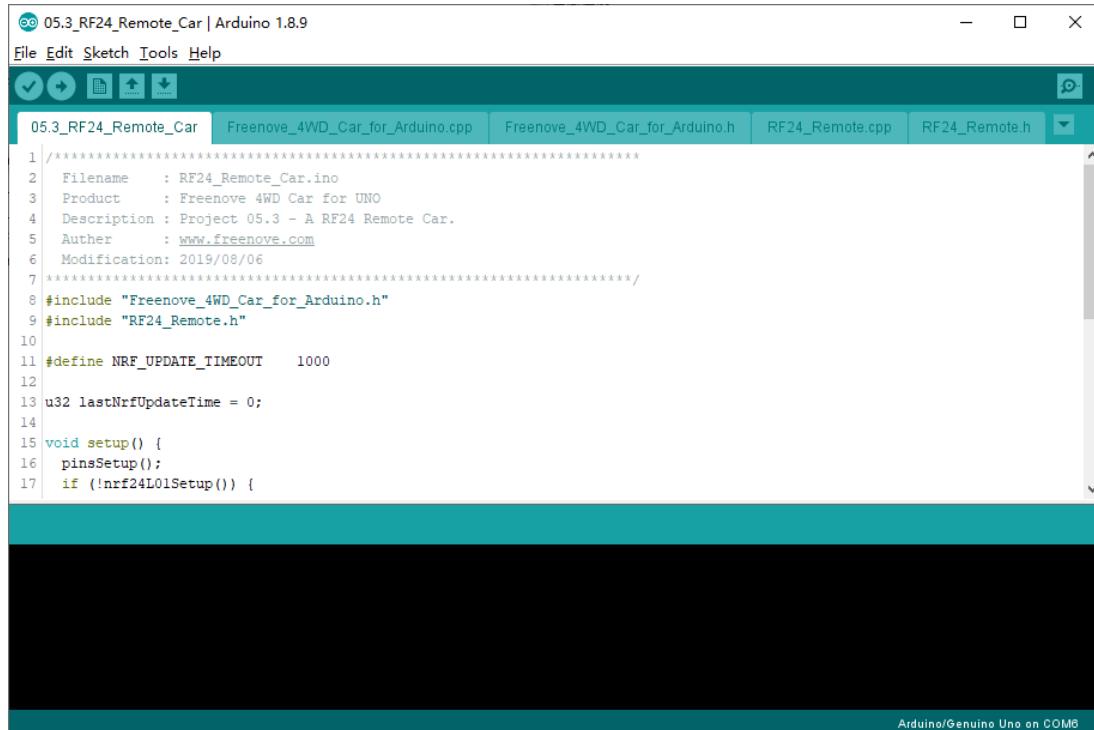
5.3 Voiture à distance RF

Après avoir reçu avec succès les données envoyées par la télécommande RF, nous pouvons utiliser les données pour contrôler le mouvement de la voiture, les sons du buzzer, etc.

Télécharger le code et exécuter

Connectez la voiture à l'ordinateur avec un câble USB et téléchargez le code dans Sketches \ 05.3_RF24_Remote_Car.ino.

Vous devez d'abord supprimer le module Bluetooth lorsque vous téléchargez le code.

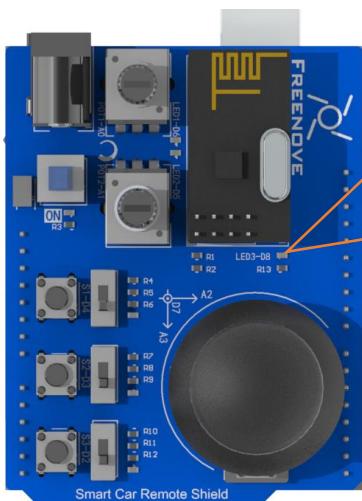


```

05.3_RF24_Remote_Car | Arduino 1.8.9
File Edit Sketch Tools Help
05.3_RF24_Remote_Car Freenove_4WD_Car_for_Arduino.cpp Freenove_4WD_Car_for_Arduino.h RF24_Remote.cpp RF24_Remote.h
1 //***** /*****
2 Filename : RF24_Remote_Car.ino
3 Product : Freenove 4WD Car for UNO
4 Description : Project 05.3 - A RF24 Remote Car.
5 Author : www.freenove.com
6 Modification: 2019/08/06
7 *****/
8 #include "Freenove_4WD_Car_for_Arduino.h"
9 #include "RF24_Remote.h"
10
11 #define NRF_UPDATE_TIMEOUT 1000
12
13 u32 lastNrfUpdateTime = 0;
14
15 void setup() {
16 pinsSetup();
17 if (!nrf24L01Setup()) {

```

Déconnectez ensuite le câble USB et allumez la télécommande RF et les interrupteurs d'alimentation du véhicule. Actionnez le joystick pour contrôler le mouvement de la voiture. Appuyez sur le joystick pour activer le buzzer.



La LED allumée indique les réussites de communication. L'état LED éteint indique que la communication échoue.

Lorsqu'il est éteint, vous devez redémarrer la télécommande et la voiture.

Code

Le projet comporte cinq étiquettes, dont "RF24_Remote.h" et "RF24_Remote.cpp" stockent le contenu sur la lecture et le traitement des données pour la télécommande RF24. Ces contenus sont appelés dans "05.3_RF24_Remote_Car.ino".

RF24_Remote.h

```
1 #ifndef _RF_REMOTE_h
2 #define _RF_REMOTE_h
3
4 # comprendre "Freenove_4WD_Car_for_Arduino.h"
5 # comprendre "RF24.h"
6
7 extern RF24 radio;
8
9 enum RemoteData {
10     POT1 = 0,
11     POT2 = 1,
12     JOYSTICK_X = 2,
13     JOYSTICK_Y = 3,
14     JOYSTICK_Z = 4,
15     S1 = 5,
16     S2 = 6,
17     S3 = 7
18 };
19
20 enum RemoteMode {
21     ON_ON_ON = 0,
22     ON_ON_OFF = 1,          // Mode d'étalonnage servo.
23     ON_OFF_ON = 2,
24     ON_OFF_OFF = 3,         // Mode d'évitement d'obstacles par ultrasons.
25     OFF_ON_ON = 4,
26     OFF_ON_OFF = 5,         // Mode de suivi de ligne.
27     OFF_OFF_ON = 6,          //
28     OFF_OFF_OFF = 7         // Mode de contrôle à distance.
29 };
30
31 enum RemoteModeSwitchState {
32     MODE_SWITCHING_IS_INITIALIZING = 0,
33     MODE_SWITCHING_IS_PROCESSING = 1,
34     MODE_SWITCHING_IS_CONFIRMING = 2,
35     MODE_SWITCHING_WAS_FINISHED = 3
36 };
37
38 #define MODE_REMOTE_CONTROL OFF_OFF_OFF
```

```

38 # définir MODE_LINE_TRACKING           OFF_ON_OFF
39 # définir MODE_OBSTACLE_AVOIDANCE      ON_OFF_OFF
40
41 externe int RFDataRead [8];
42
43 booléen RF24L01Configuration ();
44 booléen getRF24L01Data ();
45 néant updateCarActionByRFRremote ();
46 néant resetRFDataBuf ();
47 u8 updateRFCarMode ();
48
49 # fin si

```

Où enum RemoteData définit l'ordre des données de contrôle à distance reçues, en utilisant des noms de variables plus significatifs au lieu de nombres.

Par exemple, si vous souhaitez utiliser les données de l'axe X de la télécommande, vous pouvez utiliser RFDataRead [JOYSTICK_X] au lieu de RFDataRead [2] sans signification. C'est comme définir

```

enum RemoteData {
    POT1 = 0, // semblable à # définir POT1 0
    POT2 = 1,
    JOYSTICK_X = 2,
    JOYSTICK_Y = 3,
    JOYSTICK_Z = 4,
    S1 = 5,
    S2 = 6,
    S3 = 7
};

```

L'énumération RemoteMode définit tous les états des commutateurs de télécommande S1, S2, S3. L'énumération RemoteModeSwitchState définit l'état lorsque la télécommande change de mode. Ces contenus seront également utilisés dans le prochain projet.

```

enum RemoteMode {
    ON_ON_ON = 0,
    ON_ON_OFF = 1,          // Mode d'étalonnage servo.
    ON_OFF_ON = 2,
    ON_OFF_OFF = 3,         // Mode d'évitement d'obstacles par ultrasons.
    OFF_ON_ON = 4,
    OFF_ON_OFF = 5,         // Mode de suivi de ligne.
    OFF_OFF_ON = 6,          //
    OFF_OFF_OFF = 7,         // Mode de contrôle à distance.
};

enum RemoteModeSwitchState {
    MODE_SWITCHING_IS_INITIALIZING = 0,
    MODE_SWITCHING_IS_PROCESSING = 1,
}

```

```

    MODE_SWITCHING_IS_CONFIRMING = 2,
    MODE_SWITCHING_WAS_FINISHED = 3
};
```

boolén RF24L01Setup ():	
Initialisez RF24L01. En cas de succès, retournez true, sinon, retournez false. boolén	
getRF24L01Data ():	
Lisez les données de la télécommande. S'il y a des données lues, renvoie true; sinon, renvoie false. void	
updateCarActionByRFRremote ():	
Contrôlez les actions de la voiture (principalement pour le moteur et le buzzer) en fonction des données reçues. void	
resetRFDataBuf ():	
Définissez les valeurs du tableau RFData sur les valeurs par défaut. u8	
updateRFCarMode ():	
Combinez les données de S1 S2 S3 de la télécommande en une seule valeur et renvoyez-la.	

05.3_RF24_Remote_Car.ino

```

1 # comprendre "Freenove_4WD_Car_for_Arduino.h"
2 # comprendre "RF24_Remote.h"
3
4 # définir RF_UPDATE_TIMEOUT          1000
5 u32 lastRFUpdateTime = 0;
6
7 sept nant installer () {
8     brochesSetup ();
9     si (! RF24L01Configuration ()) {
dix         alarme (4, 2);
11    }
12 }
13
14 nant boucle () {
15     si ( getRF24L01Data ()) {
16         updateCarActionByRFRremote ();
17         lastRFUpdateTime = millis ();
18     }
19     si ( millis () - lastRFUpdateTime > RF_UPDATE_TIMEOUT ) {
20         lastRFUpdateTime = millis ();
21         resetRFDataBuf ();
22         updateCarActionByRFRremote ();
23     }
24 }
```

Dans setup (), initialisez les broches et RF24L01. Dans loop (), recevez les données de la télécommande et traitez les données. Si la réception des données est expirée, le signal de la télécommande est considéré comme perdu et la voiture sera placée dans l'état d'initialisation.

5.4 Voiture à distance multifonctionnelle RF24

Ce projet combine presque toutes les fonctions de la voiture, telles que le mode d'évitement d'obstacles, le mode de suivi de ligne, le mode de contrôle à distance, le mode d'affichage LED RVB, l'étalonnage de la position des servos, etc. La commutation entre différentes fonctions est réalisée en commutant les états de S1, S2 et S3 sur la télécommande.

Le code du projet est plus compliqué, mais les composants et les connaissances utilisés ont été introduits dans le projet précédent. La seule différence est qu'ils sont intégrés dans un seul projet, ce qui est assez important.

Télécharger le code et exécuter

Connectez la voiture à l'ordinateur avec un câble USB.

Vous devez d'abord supprimer le module Bluetooth lorsque vous téléchargez le code.

Et téléchargez le code dans **Croquis \ 05.5_One_Code_Multifunctional_RF24_Remote_Car**

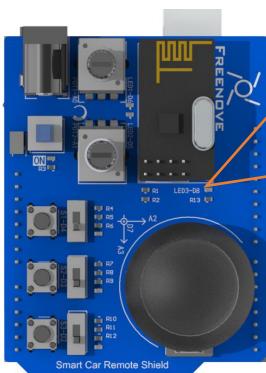
Vous pouvez également choisir Sketches \ 05.4_Multifunctional_RF24_Remote_Car.ino.

```

 05.4_Multifunctional_RF24_Remote_Car | Arduino 1.8.9
File Edit Sketch Tools Help
05.4_Multifunctional_RF24_Remote_Car Automatic_Obstacle_Avoidance_Mode.cpp Automatic_Obstacle_Avoidance_Mode.h Automatic_T king
1 // *****
2 Filename : Receive_Data.ino
3 Product : Freenove 4WD Car for UNO
4 Description : Project 05.4 - A Multifunctional RF24-Remote Car.
5 Author : www.freenove.com
6 Modification: 2019/08/08
7 *****
8 #include "Automatic_Tracking_Line_Mode.h"
9 #include "Automatic_Obstacle_Avoidance_Mode.h"
10 #include "Freenove_4WD_Car_for_Arduino.h"
11 #include "Freenove_WS2812B_RGBLED_Controller.h"
12 #include "RF24_Remote.h"
13
14 #define NRF_UPDATE_TIMEOUT 1000
15
16 u32 lastNrfUpdateTime = 0;
17 u8 nrfCarMode = OFF_OFF_OFF, lastNrfCarMode = OFF_OFF_OFF;

```

Déconnectez le câble USB. Allumez la télécommande et la voiture. Par défaut, la voiture est en mode télécommande manuelle et les commutateurs S1, S2 et S3 de la télécommande sont désactivés.



L'état LED ON indique les réussites de communication. L'état LED OFF indique que la communication échoue.

Lorsqu'il est éteint, vous devez redémarrer l'alimentation de la télécommande et de la voiture.

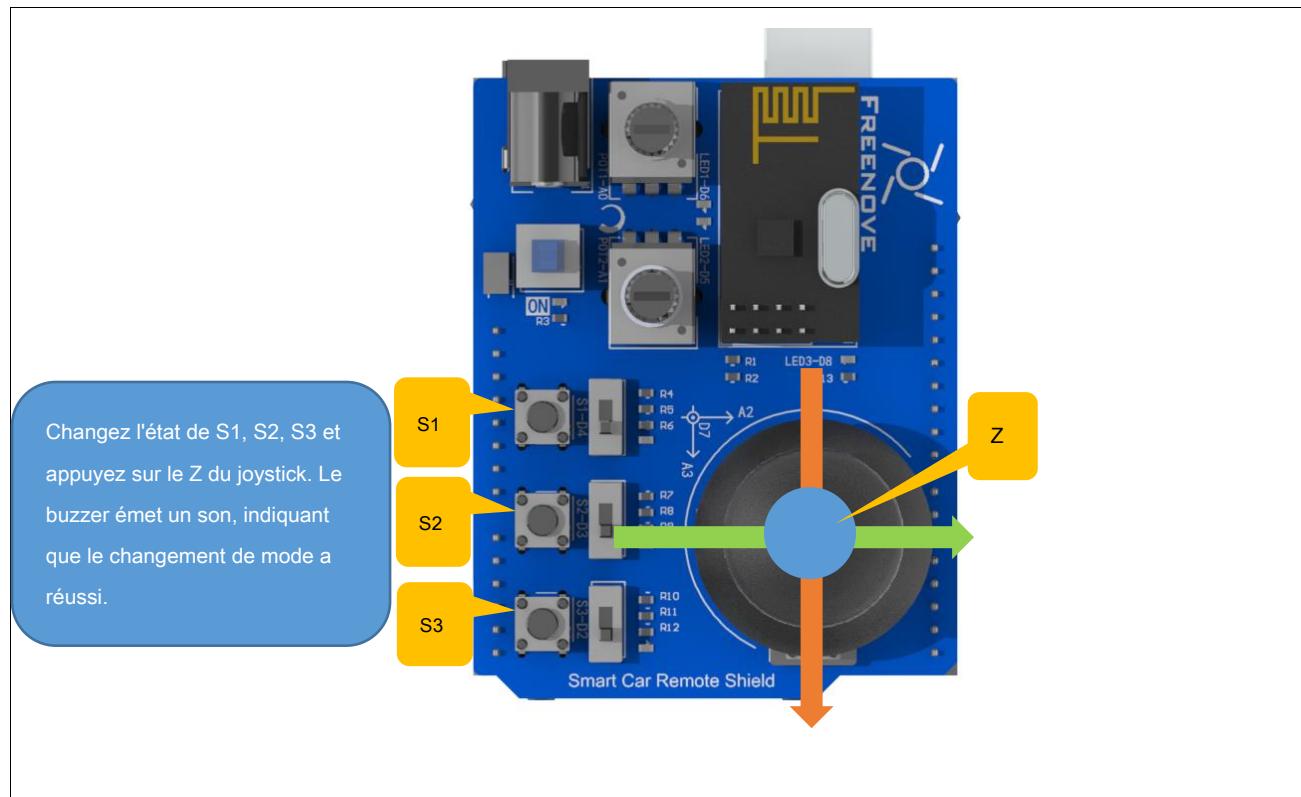
Changer de mode

1, changez l'état du commutateur de S1, S2 et S3, et la voiture cessera de bouger.

2, appuyez sur l'axe Z du joystick et le buzzer B retentit pour indiquer que le mode est commuté avec succès.

Le tableau suivant montre les modes indiqués par différents états des trois commutateurs S1, S2 et S3. La LED à côté de l'éclairage de l'interrupteur pour indiquer l'état ON et OFF des interrupteurs. Les trois commutateurs peuvent former $2 \times 2 \times 2 = 8$ modes.

S1	S2	S3	Numéro de mode	Mode
ON	ON	ON	0	Aucun
ON	ON	OFF	1	Calibrer le mode servo
ON	OFF	ON	2	Aucun
ON	OFF	OFF	3	Mode évitemen t d'obstacles
OFF	ON	ON	4	Aucun
OFF	ON	OFF	5	Mode de suivi de ligne
OFF	OFF	ON	6	Changer de mode LED
OFF	OFF	OFF	sept	Mode de contrôle manuel / Mode par défaut



Mode 0, 2, 4

Réservé. Nous ne leur avons pas attribué de fonctions.

Mode 1-Calibrer le servo

Si votre servo n'est pas monté avec précision à 90 degrés, vous pouvez utiliser ce mode pour un réglage fin (+ -10 degrés).

Dans ce mode, vous pouvez régler le potentiomètre 2 (POT2) pour affiner l'angle du servo. Lorsque vous ajustez le servo à l'angle correct, appuyez sur l'axe Z du joystick pour enregistrer les données d'étalonnage dans l'EEPROM. Il sera enregistré de manière permanente à moins qu'il ne soit modifié.

Mode 3-évitement obstical, Mode mode de suivi 5 lignes

Ces deux modes ont été appris séparément dans le projet précédent, et leur logique d'exécution et leurs codes sont cohérents avec le projet précédent.

La différence est que dans ce projet, la voiture peut répondre à tout moment aux commandes de la télécommande. Par conséquent, dans ce projet, il est toujours nécessaire de communiquer avec la télécommande dans ces deux modes. Lorsque le signal de la télécommande est déconnecté, la voiture s'arrête. Par conséquent, la communication normale entre la télécommande et la voiture doit être maintenue à tout moment. De mauvaises conditions de communication peuvent entraîner un fonctionnement anormal de ces deux modes.

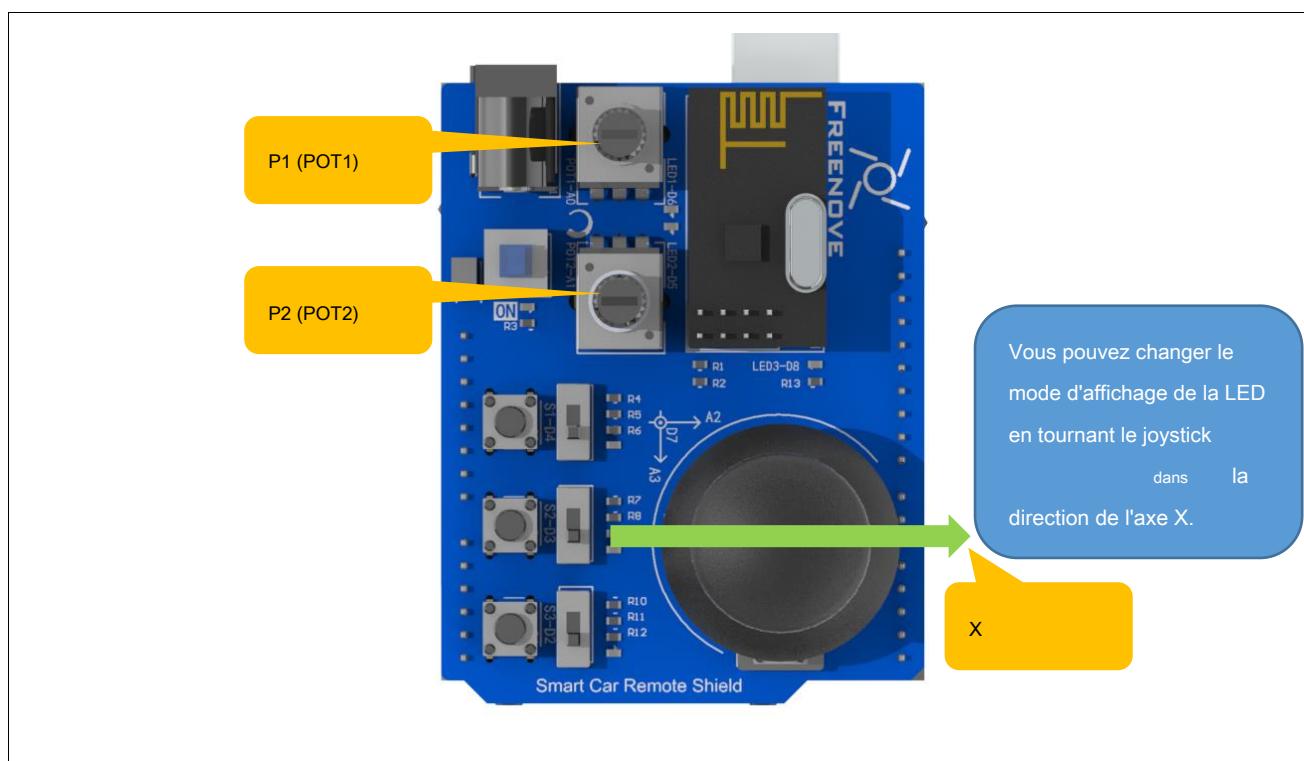
Mode d'affichage LED à 6 commutateurs

Il existe trois modes d'affichage pour les LED de la voiture, qui sont un arc-en-ciel à 0 écoulement, 1 LED à eau qui coule, 2 clignotements. Dans ce mode, le mode d'affichage de la LED peut être commuté.

Après être entré dans ce mode,

Déplacez le joystick le long de la direction positive de son axe X pour faire passer la LED au mode suivant. Déplacez le joystick dans le sens négatif de son axe X pour faire passer la LED au mode précédent.

Dans n'importe quel mode, les LED peuvent être réglées avec les potentiomètres P1 et P2. P1 est utilisé pour ajuster le changement de couleur de la LED, et P2 est utilisé pour ajuster la fréquence de changement de LED.



Mode 7 - mode manuel à distance

Ce mode est le mode manuel à distance et est le mode par défaut. Ce mode est cohérent avec le projet précédent "RF_Remote_Car". Utilisez le joystick pour contrôler pour avancer, reculer et tourner à gauche, tourner à droite.

Code

Il y a 9 étiquettes (fichier) pour ce projet, dont 8 sont apparues dans des projets précédents. Dans le fichier "05.4_Multifunctional_RF24_Remote_Car.ino", il s'agit principalement de la gestion logique de chaque fonction du module, comme la commutation du mode voiture, la commutation du mode d'affichage LED.

05.4_Multifunctional_RF24_Remote_Car.ino

```
1 # comprendre "Automatic_Tracking_Line_Mode.h"
2 # comprendre "Automatic_Obstacle_Avoidance_Mode.h"
3 # comprendre "Freenove_4WD_Car_for_Arduino.h"
4 # comprendre "Freenove_WS2812B_RGBLED_Controller.h"
5 # comprendre "RF24_Remote.h"
6
7 # définir RF_UPDATE_TIMEOUT 1000
8
9 u32 lastRFUpdateTime = 0;
dix u8 RFCarMode = OFF_OFF_OFF , lastRFCarMode = OFF_OFF_OFF ;
11 u8 switchModeState = MODE_SWITCHING_WAS_FINISHED ;
12 u8 joystickSwitchState = MODE_SWITCHING_WAS_FINISHED ;
13
14 # définir STRIP_I2C_ADDRESS 0x20
15 # définir STRIP_LEDS_COUNT dix
16
17 # définir MAX_NUMBER_OF_DISP_MODES 3
18
19 u8 colorPos = 0;
20 u8 colorStep = 50;
21 u8 stripDisplayMode = 1;
22 u8 currentLedIndex = 0;
23 u16 stripDisplayDelay = 100;
24 u32 lastStripUpdateTime = 0;
25 Freenove_WS2812B_Controller bande ( STRIP_I2C_ADDRESS , STRIP_LEDS_COUNT , TYPE_GRB );
26
27 néant installer () {
28     brochesSetup ();
29     si (! RF24L01Configuration ()) {
30         alarme (4, 2);
31     }
32     servoSetup ();
33     tandis que (! bande . commencer ());
34     bande . setAllLedsColor (0xFF0000);
35 }
36
37 néant boucle () {
```

```

38     si ( getRF24L01Data () ) {
39         RFCarMode = updateRFCarMode ();
40         si ( RFCarMode != lastRFCarMode ) {
41             si ( switchModeState == MODE_SWITCHING_WAS_FINISHED ) {
42                 switchModeState = MODE_SWITCHING_IS_INITIALIZING ;
43             }
44             commutateur ( switchModeState )
45             {
46                 Cas MODE_SWITCHING_IS_INITIALIZING :
47                     //Serial.println("Initialisation du mode de commutation ... ");
48                     resetCarAction ();
49                     writeServo (90);
50                     switchModeState = MODE_SWITCHING_IS_PROCESSING ;
51                     Pause ;
52                 Cas MODE_SWITCHING_IS_PROCESSING :
53                     si ( RFDataRead [ JOYSTICK_Z ] == 0 ) {
54                         //Serial.println("Mode de commutation ... ");
55                         setBuzzer ( vrai );
56                         switchModeState = MODE_SWITCHING_IS_CONFIRMING ;
57                     }
58                     Pause ;
59                 Cas MODE_SWITCHING_IS_CONFIRMING :
60                     si ( RFDataRead [ JOYSTICK_Z ] == 1 ) {
61                         //Serial.println("Comfirm Commuté. ");
62                         setBuzzer ( faux );
63
64                         switchModeState = MODE_SWITCHING_WAS_FINISHED ;
65                         lastRFCarMode = RFCarMode ;
66                         commutateur ( RFCarMode )
67                         {
68                             Cas MODE_OBSTACKE_AVOIDANCE :
69                                 oa_CalculateVoltageCompensation ();
70                                 Pause ;
71                             Cas MODE_LINE_TRACKING :
72                                 tk_CalculateVoltageCompensation ();
73                                 Pause ;
74                             défaut :
75                                 Pause ;
76                         }
77                         //Serial.println("Mode de commutation terminé! ");
78                     }
79                     Pause ;
80                 Cas MODE_SWITCHING_WAS_FINISHED :
81                     Pause ;

```

```

82         défaut :
83             Pause ;
84         }
85     }
86     autre {
87         si ( switchModeState != MODE_SWITCHING_WAS_FINISHED ){
88             switchModeState = MODE_SWITCHING_WAS_FINISHED ;
89         }
90         commutateur ( RFCarMode )
91         {
92             Cas ON_ON_ON :
93                 Pause ;
94             Cas ON_ON_OFF :      // S1, S2 ON, S3 OFF
95                 setServoOffset ( carte ( nrfDataRead [1], 0, 1023, -20, 20));
96                 si ( RFDataRead [ JOYSTICK_Z ] == 0 ){
97                     setBuzzer ( vrai );
98                     writeServoOffsetToEEPROM ();
99                 }
100            autre {
101                setBuzzer ( faux );
102            }
103            Pause ;
104            Cas ON_OFF_ON :
105                Pause ;
106            Cas ON_OFF_OFF :      //// Mode d'évitement d'obstacle sonique, S1 est activé et S2, S3 sont
107 DE
108             updateAutomaticObstacleAvoidance ();
109             Pause ;
110             Cas OFF_ON_ON :
111                 Pause ;
112             Cas OFF_ON_OFF :      // Mode de suivi, S2 est activé et S1, S3 sont désactivés.
113                 updateAutomaticTrackingLine ();
114                 Pause ;
115             Cas OFF_OFF_ON :      // S3 est ON et S1, S2 sont OFF
116             // stripDisplayMode = RFDataRead [POT2] / 512;
117             commutateur ( joystickSwitchState )
118             {
119                 statique u8 switchCounter = 0;
120                 Cas MODE_SWITCHING_IS_INITIALIZING :
121                     si ( RFDataRead [ JOYSTICK_X ] > 900 ){
122
123                         setBuzzer ( vrai );
124                         switchCounter++;
125                         joystickSwitchState = MODE_SWITCHING_IS_PROCESSING ;

```

```
126 }
127         }
128
129         sinon si ( RFDataRead [ JOYSTICK_X ] <200 ) {
130             setBuzzer ( vrai );
131             switchCounter -;
132             joystickSwitchState = MODE_SWITCHING_IS_PROCESSING ;
133         }
134         Pause ;
135         Cas MODE_SWITCHING_IS_PROCESSING :
136             si (( RFDataRead [ JOYSTICK_X ] <600) && ( RFDataRead [ JOYSTICK_X ] > 400)) {
137                 setBuzzer ( faux );
138                 joystickSwitchState = MODE_SWITCHING_IS_CONFIRMING ;
139             }
140             Pause ;
141             Cas MODE_SWITCHING_IS_CONFIRMING :
142                 stripDisplayMode += switchCounter ;
143                 si ( stripDisplayMode == 0xff) {
144
145                     stripDisplayMode = MAX_NUMBER_OF_DISP_MODES - 1;
146                 }
147                 stripDisplayMode %= MAX_NUMBER_OF_DISP_MODES ;
148                 joystickSwitchState = MODE_SWITCHING_WAS_FINISHED ;
149                 Pause ;
150                 Cas MODE_SWITCHING_WAS_FINISHED :
151                     switchCounter = 0;
152                     joystickSwitchState = MODE_SWITCHING_IS_INITIALIZING ;
153                     Pause ;
154                     défaut :
155                         Pause ;
156                     }
157                     Pause ;
158                     Cas OFF_OFF_OFF :           // Mode à distance, tous les voyants de commutation sont éteints
159                         updateCarActionByRFRemote ();
160                         Pause ;
161                         défaut :
162                             Pause ;
163                         }
164                     }
165                     colorStep = carte ( RFDataRead [ POT1 ], 0, 1023, 0, 255);
166                     stripDisplayDelay = RFDataRead [ POT2 ];
167                     lastRFUpdateTime = millis ();
168                 }
169 }
```

```
170     si ( millis () - lastRFUpdateTime > RF_UPDATE_TIMEOUT ){
171         lastRFUpdateTime = millis ();
172         resetRFDataBuf ();
173         updateCarActionByRFRremote ();
174         RFCarMode = lastRFCarMode = MODE_REMOTE_CONTROL ;
175     }
176     commutateur ( stripDisplayMode )
177     {
178         Cas 0:
179         si ( millis () - lastStripUpdateTime > stripDisplayDelay ) {
180
181             pour ( int je = 0; je < STRIP_LEDS_COUNT ; je ++){
182                 bande . setLedColorData ( je , bande . Roue ( colorPos + je * 25));
183             }
184             bande . montrer ();
185             colorPos += colorStep ;
186             lastStripUpdateTime = millis ();
187         }
188         Pause ;
189     Cas 1:
190     si ( millis () - lastStripUpdateTime > stripDisplayDelay ) {
191
192         bande . setLedColor ( currentLedIndex , bande . Roue ( colorPos ));
193         currentLedIndex++;
194         si ( currentLedIndex == STRIP_LEDS_COUNT ){
195
196             currentLedIndex = 0;
197             colorPos += colorStep ; //
198         }
199         lastStripUpdateTime = millis ();
200     }
201     Pause ;
202     Cas 2:
203     colorPos = colorStep ;
204     si ( millis () - lastStripUpdateTime > stripDisplayDelay ) {
205
206         statique booléen ledState = vrai ;
207         si ( ledState )
208         {
209             bande . setAllLedsColor ( bande . Roue ( colorPos ));
210         }
211         autre
212         {
213             bande . setAllLedsColor ( 0x00);
```

```

214         }
215         ledState =! ledState ;
216         lastStripUpdateTime = millis ();
217     }
218     Pause ;
219     défaut :
220     Pause ;
221 }
222 }
```

Dans setup (), initialisez les broches et le RF24L01 et le module LED. Dans loop (), recevez les données de la télécommande et traitez les données. Si la réception des données a expiré, le signal de la télécommande est considéré comme perdu. Affichage LED

code est ajouté dans ce code, qui est différent du code du projet précédent.

```

nément installer () {
    brochesSetup ();
    si (! RF24L01Configuration ()) {
        alarme (4, 2);
    }
    servoSetup ();
    tandis que (! bande . commencer ());
        bande . setAllLedsColor (0xFF0000);
    }

nément boucle () {
    si ( getRF24L01Data ()) {
        .....
    }

    si ( millis () - lastRFUpdateTime > RF_UPDATE_TIMEOUT ) {
        lastRFUpdateTime = millis ();
        resetRFDataBuf ();
        updateCarActionByRFRremote ();
        RFCarMode = lastRFCarMode = MODE_REMOTE_CONTROL ;
    }
    commutateur ( stripDisplayMode )
    {
    .....
    }
}
```

Après avoir lu les données de la télécommande, jugez si le mode de la voiture a changé en fonction de l'état de S1, S2, S3. Si le mode est changé, le code du mode correspondant sera exécuté. Sinon, le subs le code suivant sera exécuté.

```
si ( getRF24L01Data () {
    RFCarMode = updateRFCarMode ();
    si ( RFCarMode != lastRFCarMode ) {
        si ( switchModeState == MODE_SWITCHING_WAS_FINISHED ) {
            switchModeState = MODE_SWITCHING_IS_INITIALIZING ;
        }
        commutateur ( switchModeState )
        {
            .....
        }
        .....
    }
    autre {
        .....
    }
}
```

Lorsque l'état de S1, S2, S3 change, cela signifie que l'utilisateur veut changer le mode de la voiture, mais afin de s'assurer que ce changement n'est pas une opération erronée, l'utilisateur doit appuyer sur l'axe Z du joystick pour confirmer.

Cette série d'actions peut être mise en œuvre en utilisant l'idée d'une machine à états finis. Lorsque l'état du commutateur change, l'état "MODE_SWITCHING_IS_INITIALIZING" est entré, le module d'alimentation, le moteur et le servo de la voiture sont initialisés pour empêcher la voiture de fonctionner de manière incontrôlée.

```
Cas MODE_SWITCHING_IS_INITIALIZING :
//Serial.println("Initialisation du mode de commutation ... ");
resetCarAction ();
writeServo (90);
switchModeState = MODE_SWITCHING_IS_PROCESSING ;
Pause ;
```

Entrez ensuite l'état suivant "MODE_SWITCHING_IS_PROCESSING" et attendez que l'utilisateur appuie sur l'axe Z de la bascule pour confirmer. Si l'utilisateur appuie sur l'axe Z du joystick, le buzzer retentit et entre dans l'état suivant "MODE_SWITCHING_IS_CONFIRMING", en attendant que l'utilisateur relâche le joystick. Lorsque l'utilisateur relâche le joystick, le buzzer cesse de sonner et change la valeur de la variable switchModeState qui contient le mode voiture. Une fois le mode commuté, l'état d'achèvement "MODE_SWITCHING_WAS_FINISHED" est entré.

```
Cas MODE_SWITCHING_IS_PROCESSING :
si ( RFDataRead [ JOYSTICK_Z ] == 0 {
    //Serial.println("Mode de commutation ... ");
    setBuzzer ( vrai );
    switchModeState = MODE_SWITCHING_IS_CONFIRMING ;
}
Pause ;
```

```

Cas MODE_SWITCHING_IS_CONFIRMING :
    si ( RFDataRead [ JOYSTICK_Z ] == 1 ) {
        //Serial.println("Comfirm Commuté. ");
        setBuzzer ( faux );
        switchModeState = MODE_SWITCHING_WAS_FINISHED ;
        lastRFCarMode = RFCarMode ;
        commutateur ( RFCarMode )
    }

    Cas MODE_OBSTACLE_AVOIDANCE :
        oa_CalculateVoltageCompensation ();
        Pause ;

    Cas MODE_LINE_TRACKING :
        tk_CalculateVoltageCompensation ();
        Pause ;

    défaut :
        Pause ;
    }

    //Serial.println("Mode de commutation terminé! ");
}

Pause ;

Cas MODE_SWITCHING_WAS_FINISHED :
    Pause ;
défaut :
    Pause ;
}

```

Ensuite, utilisez l'instruction switch-case pour laisser la voiture exécuter la fonction ou l'action correspondante en fonction de la valeur de la variable RFCarMode de mode voiture.

```

commutateur ( RFCarMode )

{
    Cas ON_ON_ON :
        Pause ;

    Cas ON_ON_OFF :      // S1, S2 ON, S3 OFF
        ..... // Définir le décalage du servo

    Cas ON_OFF_ON :
        Pause ;

    Cas ON_OFF_OFF : //// Mode d'évitement d'obstacle sonique, S1 est activé et S2, S3 sont désactivés
        updateAutomaticObstacleAvoidance ();
        Pause ;

    Cas OFF_ON_ON :
        Pause ;

    Cas OFF_ON_OFF :      // Mode de suivi, S2 est activé et S1, S3 sont désactivés.
        updateAutomaticTrackingLine ();
        Pause ;
}

```

```

Cas OFF_OFF_ON :           // S3 est ON et S1, S2 sont OFF
..... // Changer le mode LED

Pause ;

Cas OFF_OFF_OFF :         // Mode à distance, tous les voyants de commutation sont éteints
updateCarActionByRFRemote ();

Pause ;

défaut :

Pause ;

}

```

En mode servo d'étalementage, l'angle du servo est ajusté en fonction de la valeur du potentiomètre POT2. Lorsque vous appuyez sur l'axe Z de la bascule, les données ajustées sont écrites dans l'EEPROM et le buzzer faire un son pour confirmer l'écriture.

```

Cas ON_ON_OFF :           // S1, S2 ON, S3 OFF
setServoOffset ( RFDataRead [1100];
si ( RFDataRead [ JOYSTICK_Z ] == 0 ) {
    setBuzzer ( vrai );
    writeServoOffsetToEEPROM ();
}
autre {
    setBuzzer ( faux );
}
Pause ;

```

Dans le mode de commutation de l'affichage LED, l'idée de machine à états finis est toujours utilisée.

Tout d'abord, entrez dans l'état d'initialisation, en attendant que le joystick se déplace dans la direction de l'axe X. Si tel est le cas, la variable switchCounter sera auto-ajoutée ou auto-décrémentée selon les directions positive et négative. Entrez ensuite dans l'état suivant, attendez que le joystick revienne au milieu. Lorsque le joystick revient au milieu, entrez dans l'état suivant, traitez les données, une fois le traitement des données terminé, entrez l'achèvement

Etat.

```

commutateur ( joystickSwitchState )
{
    statique u8 switchCounter = 0;
Cas MODE_SWITCHING_IS_INITIALIZING :
    si ( RFDataRead [ JOYSTICK_X ] > 900 ) {

        setBuzzer ( vrai );
        switchCounter++;
        joystickSwitchState = MODE_SWITCHING_IS_PROCESSING ;
    }
    sinon si ( RFDataRead [ JOYSTICK_X ] < 200 ) {

        setBuzzer ( vrai );
        switchCounter--;
    }
}

```

```

        joystickSwitchState = MODE_SWITCHING_IS_PROCESSING ;
    }

    Pause ;

Cas MODE_SWITCHING_IS_PROCESSING :
    si (( RFDataRead [ JOYSTICK_X ] <600 ) && ( RFDataRead [ JOYSTICK_X ] > 400 )) {

        setBuzzer ( faux );
        joystickSwitchState = MODE_SWITCHING_IS_CONFIRMING ;
    }

    Pause ;

Cas MODE_SWITCHING_IS_CONFIRMING :
    stripDisplayStyle += switchCounter ;
    si ( stripDisplayStyle == 0xff ) {

        stripDisplayStyle = MAX_NUMBER_OF_DISP_MODES - 1;
    }

    stripDisplayStyle %= MAX_NUMBER_OF_DISP_MODES ;
    joystickSwitchState = MODE_SWITCHING_WAS_FINISHED ;
    Pause ;

Cas MODE_SWITCHING_WAS_FINISHED :
    switchCounter = 0;
    joystickSwitchState = MODE_SWITCHING_IS_INITIALIZING ;
    Pause ;
    défaut :
        Pause ;
    }
    Pause ;
}

```

La variable qui contrôle le changement de couleur de l'affichage LED et la variable de la fréquence d'affichage sont mises à jour chaque temps selon les données de la télécommande.

```

colorStep = carte ( RFDataRead [ POT1 ], 0, 1023, 0, 255);
stripDisplayDelay = RFDataRead [ POT2 ];

```

Enfin, la variable de mode d'affichage LED stripDisplayStyle, la variable d'amplitude de changement de couleur colorStep, la afficher y période variable stripDisplayDelay.

```

commutateur ( stripDisplayStyle )
{
    Cas 0:
        si ( millis () - lastStripUpdateTime > stripDisplayDelay ) {

            pour ( int je = 0; je < STRIP_LEDS_COUNT ; je ++ ) {
                bande . setLedColorData ( je , bande . Roue ( colorPos + je * 25));
            }
            bande . montrer ();
        }
}

```

```
colorPos += colorStep ;
lastStripUpdateTime = millis ();
}

Pause ;

Cas 1:
si ( millis () - lastStripUpdateTime > stripDisplayDelay ) {

    bande . setLedColor ( currentLedIndex , bande . Roue ( colorPos ));
    currentLedIndex++;
    si ( currentLedIndex == STRIP_LEDS_COUNT ) {

        currentLedIndex = 0;
        colorPos += colorStep ; //
    }
    lastStripUpdateTime = millis ();
}
}

Pause ;

Cas 2:
colorPos = colorStep ;
si ( millis () - lastStripUpdateTime > stripDisplayDelay ) {

    statique booléen ledState = vrai ;
    si ( ledState )
    {
        bande . setAllLedsColor ( bande . Roue ( colorPos ));
    }
    autre
    {
        bande . setAllLedsColor (0x00);
    }
    ledState =! ledState ;
    lastStripUpdateTime = millis ();
}
}

Pause ;

défaut :
Pause ;
}
```

Chapitre 6 Contrôle Bluetooth

Cette section nécessite un appareil Android ou iPhone avec Bluetooth pour contrôler la voiture. Si vous avez des préoccupations, n'hésitez pas à nous contacter via support@freenove.com

6.1 Configurer Bluetooth et recevoir des données

Application iOS pour Android et iPhone

Téléchargez et installez l'application. Vous pouvez utiliser l'application pour contrôler le robot. Vous pouvez télécharger l'application Freenove de différentes manières: Afficher ou télécharger sur Google Play:

<https://play.google.com/store/apps/details?id=com.freenove.suhayl.Freenove>

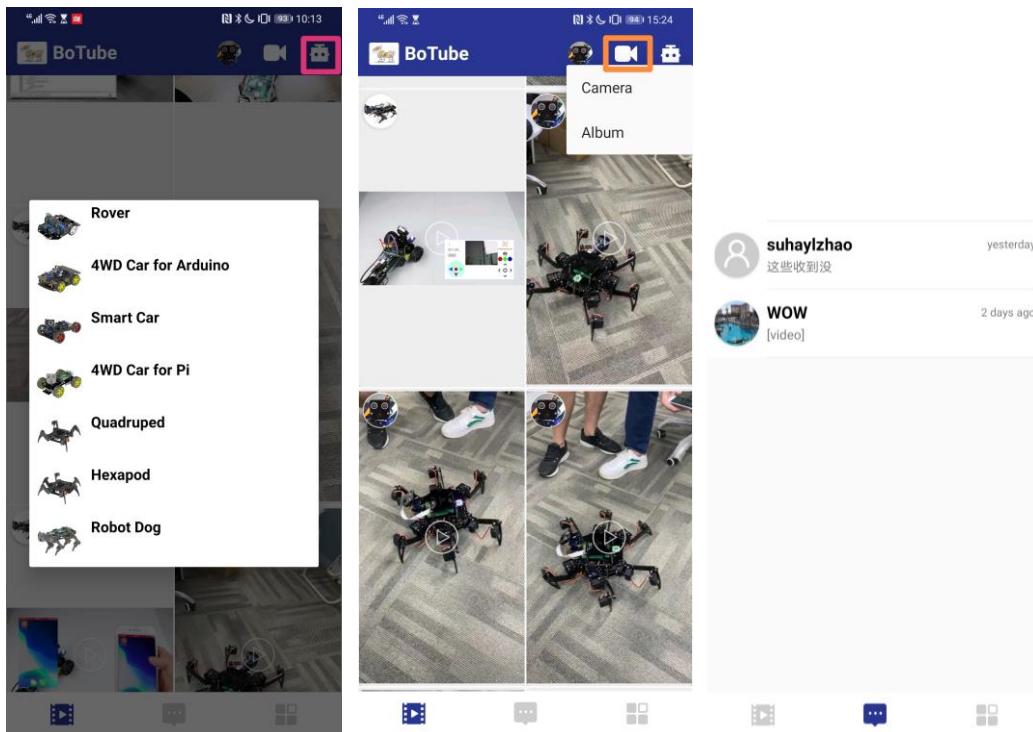
Télécharger le fichier APK directement

https://github.com/Freenove/Freenove_App_for_Android/raw/master/freenove.apk

Ensuite, installez-le sur votre téléphone Android.

Vous pouvez également utiliser l'application suivante pour contrôler cette voiture. En outre, il existe des vidéos sur les projets de robot partagés par d'autres sur cette application, ainsi que des vidéos des produits de notre entreprise. Vous pouvez également partager vos propres vidéos sur cette application. De plus, vous pouvez également discuter avec d'autres utilisateurs sur cette vidéo. Vous êtes invités à utiliser cette application. Vous pouvez le télécharger sur google play:

<https://play.google.com/store/apps/details?id=com.robotech.boogoo>



Pour application iPhone, s'il te plait serach freenove dans Magasin d'applications.



Instructions d'utilisation du module Bluetooth

Le module Bluetooth utilise le port série pour communiquer avec l'Arduino. La communication série est également utilisée pour télécharger le programme sur Arduino. **Par conséquent, lors du téléchargement du programme sur Arduino, le module Bluetooth doit être débranché; sinon, le téléchargement du programme échouera!**

Lorsque le module Bluetooth n'est pas connecté par un autre appareil, le module Bluetooth peut être configuré à l'aide de la commande AT. Une fois connecté, le module Bluetooth agit comme un canal de données et ne peut pas être configuré.

Par défaut, le module Bluetooth a une vitesse de transmission de 9600, pas de parité, 8 bits de données et 1 bit d'arrêt. Nom Bluetooth "BT05", le mode rôle est le mode esclave.

Définir le nom du bluetooth

Si vous avez plusieurs modules Bluetooth avec le même nom autour de vous, vous serez confus lorsque vous vous connectez. À quel module Bluetooth je souhaite me connecter?

Dans le prochain projet, nous présenterons comment utiliser la commande AT pour modifier le nom du module Bluetooth et le rôle maître-esclave dans le programme.

Pour les commandes AT et plus d'informations sur le module Bluetooth, reportez-vous à la documentation dans le package pour Datasheets / BT05-Instruction.pdf

Ce projet utilise la commande AT pour définir certains paramètres du module Bluetooth, puis utilise le module Bluetooth pour recevoir les données de l'application et imprimer les données sur le moniteur série.

Télécharger le code et exécuter

Vous devez d'abord supprimer le module Bluetooth lorsque vous téléchargez le code.

Ensuite, téléchargez le code dans Sketches / 06.1_Receive_Bluetooth_Data.ino.

Notez que lors du téléchargement du code, nous devons d'abord débrancher Bluetooth. Ou le téléchargement échouera.

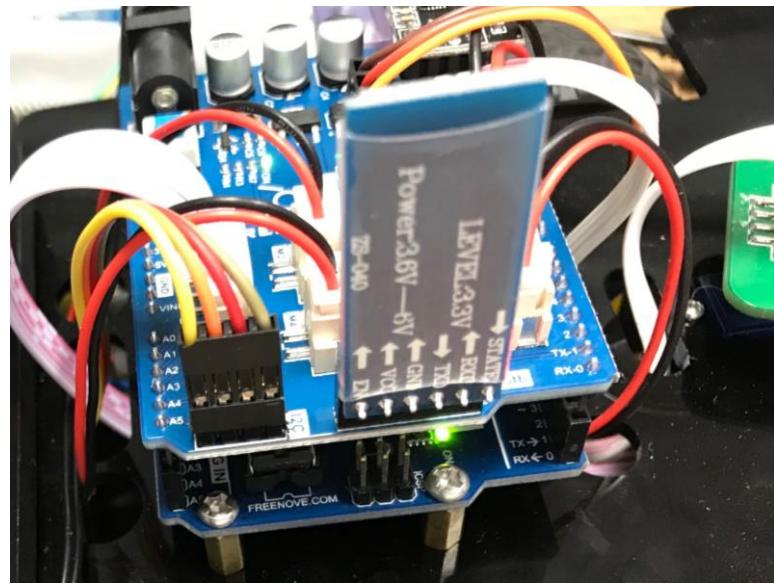
The screenshot shows the Arduino IDE interface. The title bar says "06.1_Receive_Bluetooth_Data | Arduino 1.8.9". The menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu is a toolbar with icons for upload, refresh, and other functions. The main code editor window displays the following code:

```
06.1_Receive_Bluetooth_Data
1 //*****
2 * Filename : Receive_Bluetooth_Data.ino
3 * Product : Freenove 4WD Car for UNO
4 * Description : Project 06.1 - Receive data from bluetooth and print it to monitor.
5 * Author : www.freenove.com
6 * Modification: 2019/08/06
7 * Notes : This code comes from the sample program SerialEvent.ino.
8 *****/
9 String inputString = ""; // a String to hold incoming data
10 bool stringComplete = false; // whether the string is complete
11
12 void setup() {
13 // initialize serial:
14 Serial.begin(9600);
15 Serial.println("AT+NAMEBT05");
16 delay(200);
17 Serial.println("AT+ROLE0");
18 delay(200);
19 // reserve 200 bytes for the inputString:
20 inputString.reserve(200);
21 }
```

The status bar at the bottom shows "Done uploading." and "avrduke done. Thank you." The bottom right corner indicates "Arduino/Genuino Uno en COM6".

Une fois le code téléchargé, suivez les étapes ci-dessous.

1. Branchez le module Bluetooth sur la voiture comme indiqué ci-dessous. **Ne l'inversez pas. Une mauvaise connexion peut endommager votre matériel.**



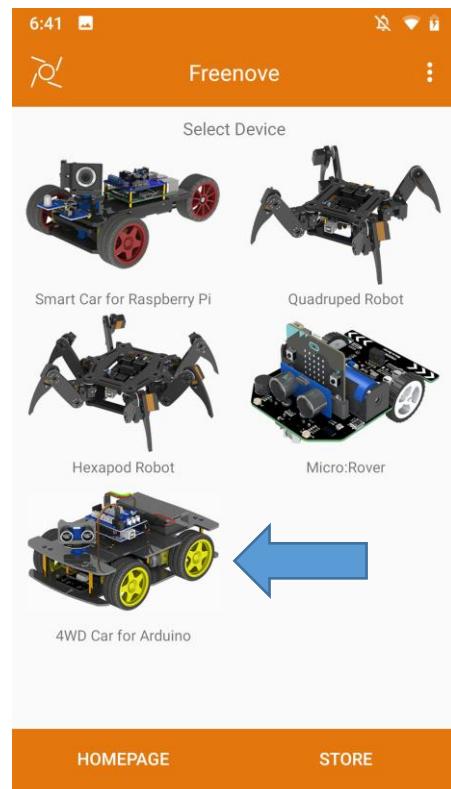
2. Ouvrez le moniteur Arduino.

Comme le montre la figure ci-dessous, le nom Bluetooth est réglé sur «BT05» et le mode Bluetooth est le mode esclave (ROLE = 0).

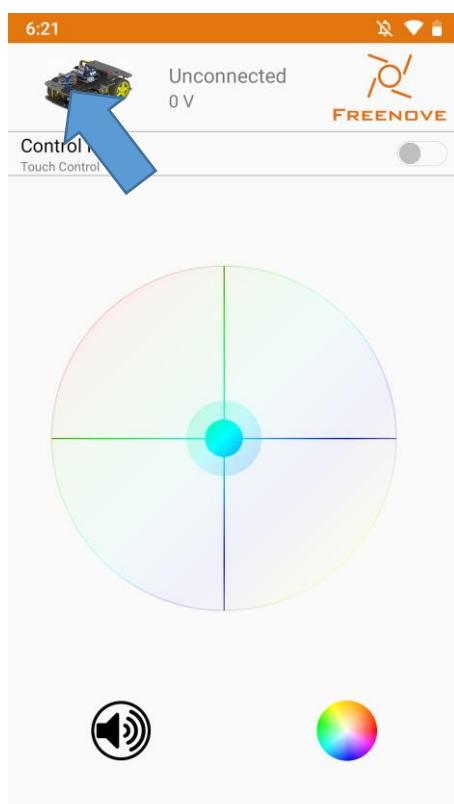
```
AT+NAMEBT05
AT+ROLE0
+NAME=BT05
OK
+ROLE=0
OK
```

Autoscroll Show timestamp Carriage return 9600 baud Clear output

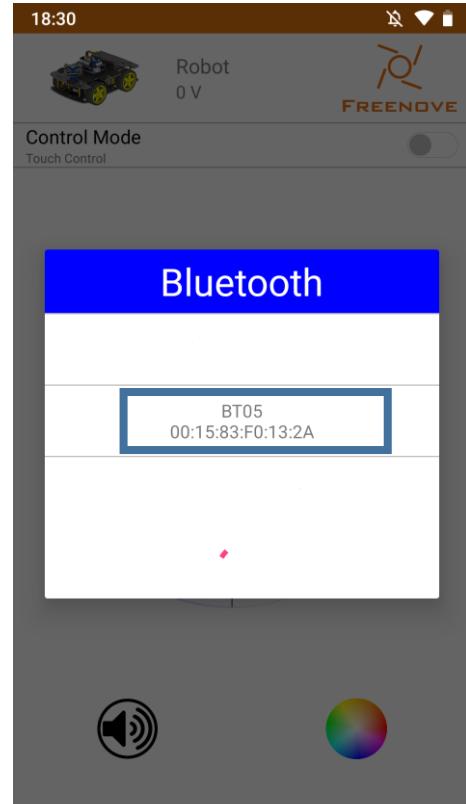
Ensuite, ouvrez l'application et cliquez sur la voiture 4WD pour Arduino.



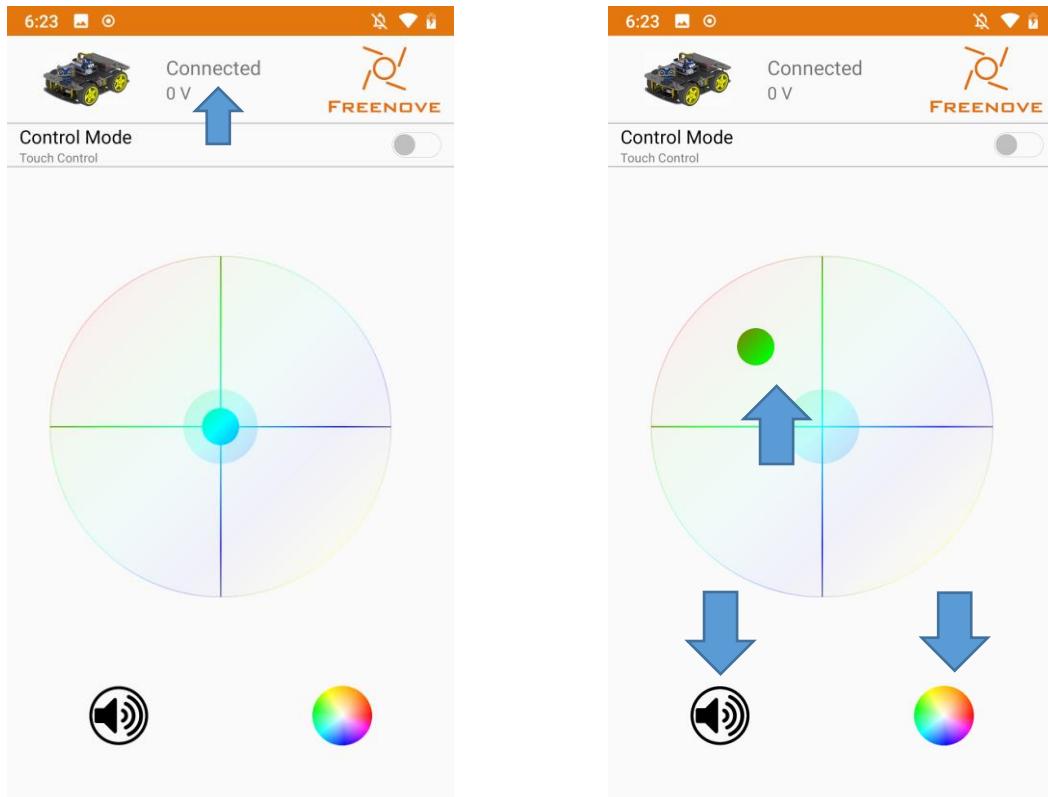
Cliquez sur l'icône suivante.



Cliquez sur BT05.



Si la connexion réussit, elle sera connectée.



Le moniteur affichera le contenu ci-dessous:

```

COM55
A#262#210#
A#0#0#
A#-248#-214#
A#0#0#
D#2000#
D#0#
C#0#189#199#255#
C#0#189#199#255#

```

Autoscroll Show timestamp Newline 9600 baud Clear output

Données Bluetooth - action de voiture

Le format de commande pour la communication entre l'application et la voiture est A # xxx # xxx # ... xxx #, où # est un séparateur, le premier caractère A représente la commande d'action, il peut s'agir d'autres caractères, tels que B, C, D ... Le xxx représente les paramètres de la commande d'action. Et différentes commandes portent des paramètres différents. La liste des commandes est la suivante:

action commander	La description	Commander personnage	Nombre de paramètres (application envoyer recevoir)	Exemple de format (envoi / réception d'application)
BOUGE TOI	Déplacer, les paramètres sont les vitesses des deux moteurs	UNE	2	Un # 100 # 100 #
ARRÊTEZ	Arrête de bouger	B	0	B #
LED_RGB	Contrôle RGBLED, paramètres sont respectivement le numéro de série du mode LED, la valeur rouge, la valeur verte, la valeur bleue.	C	3	C # 2 # 100 # 150 # 200 #
AVERTISSEUR SONORE	Activer de contrôle, et le paramètre est la fréquence.	ré	1	D # 2000 #
TENSION	Avoir la tension des batteries, le paramètre est la valeur de tension, unité mV	je	0/1	Je # / I # 4100 #

Code

06.1_Receive_Bluetooth_Data

Les données du module Bluetooth sont envoyées à l'Arduino via le port série. Le code du projet provient de l'exemple de code Arduino "SerialEvent". Recevez les données du port série et imprimez les données lorsque «\ n» est reçu.

Le code est ci-dessous:

```

1  Chaîne inputString = ""; // une chaîne pour contenir les données entrantes
2  boolen stringComplete = faux; // si la chaîne est complète
3
4  néant installer () {
5      // initialise série:
6      En série . commencer (9600); // la vitesse de transmission de données par défaut du module Bluetooth est de 9600
7      En série . println ( "AT + NAMEBT05" ); // Ou utilisez Serial.print ("AT + NAMEBT05 \ r \ n");
8      retard (200);
9      En série . println ( "AT + ROLE0" );
10     retard (200);
11     // réserve 200 octets pour la chaîne inputString:
12     inputString . réserve (200);
13 }
14

```

```

15 néant boucle () {
16     // imprime la chaîne à l'arrivée d'une nouvelle ligne:
17     si ( stringComplete ) {
18         En série . println ( inputString );
19         // efface la chaîne:
20         inputString = "";
21         stringComplete = faux ;
22     }
23 }
24
25 /*
26 SerialEvent se produit chaque fois qu'une nouvelle donnée arrive dans le RX série matériel. Cette routine est exécutée entre
27 chaque exécution de time loop (), donc l'utilisation de delay inside loop peut retarder la réponse. Plusieurs octets de données
28 peuvent être disponibles.
29 */
30 néant serialEvent () {
31     tandis que ( En série . disponible () ) {
32         // récupère le nouvel octet:
33         carboniser inChar = ( carboniser ) En série . lis ();
34         // ajoutez-le à inputString:
35         inputString += inChar ;
36         // si le caractère entrant est une nouvelle ligne, définissez un drapeau pour que la boucle principale // puisse y faire
37         quelque chose:
38         si ( inChar == '\n' ) {
39             stringComplete = vrai ;
40         }
41     }
42 }

```

La vitesse de transmission de données par défaut du module Bluetooth est de 9600, réglez donc la vitesse de transmission du port série sur 9600.

	Serial.begin (9600); // la vitesse de transmission de données par défaut du module Bluetooth est de 9600
--	--

Utilisez ensuite la commande AT pour définir le nom du module Bluetooth sur "BT05". Le format de la commande est "AT + NAMExxx \r\n", où "AT + NAME" est un format fixe et le "xxx" suivant est le nom de l'ensemble. La longueur maximale du nom est de 18 caractères. La commande doit être suivie de "\r\n" se termine. Puisque la fonction Serial.println () ajoute "\r\n" à la fin, il n'est pas nécessaire de l'ajouter dans le programme. Le délai est de 200 ms pour garantir que le module Bluetooth dispose de suffisamment de temps pour terminer la configuration.

	En série . println ("AT + NAMEBT05"); // Ou utilisez Serial.print ("AT + NAMEBT05 \r\n"); retard (200);
--	---

De même, continuez à utiliser la commande AT pour définir le mode de rôle du module Bluetooth en mode esclave. Le module Bluetooth peut être réglé en mode maître ou en mode esclave. En mode maître, le module Bluetooth peut rechercher activement et se connecter à d'autres appareils Bluetooth, mais ne peut pas être recherché et connecté par d'autres appareils. Pour être recherché par l'application mobile via Bluetooth, le module de rôle Bluetooth doit être réglé en mode esclave. Selon le manuel du fabricant Bluetooth, 0 est le mode esclave et 1 est le maître

mode.

```
En série . println ( "AT + ROLE0" );
retard (200);
```

Dans loop (), continuez à interroger si stringComplete est vrai. Si c'est vrai, cela indique que les données complètes ont été reçues. Les données seront imprimées sur le moniteur série, puis les données sont effacées. Et puis le tableau est prêt pour recevoir de nouvelles données.

```
si ( stringComplete ) {
    En série . println ( inputString );
    // efface la chaîne:
    inputString = "";
    stringComplete = faux ;
}
```

La fonction serialEvent () est utilisée pour recevoir les données transmises par le port série. Lorsque le caractère de nouvelle ligne est recevoir ed, l'indicateur stringComplete est mis à true, et les données reçues seront imprimées dans loop () .

```
néant serialEvent () {
    tandis que ( En série . disponible () ) {
        // récupère le nouvel octet:
        carboniser inChar = ( carboniser ) En série . lis ();
        // ajoutez-le à inputString:
        inputString += inChar ;
        // si le caractère entrant est une nouvelle ligne, définissez un drapeau pour que la boucle principale // puisse y faire
        quelque chose:
        si ( inChar == '\n' ) {
            stringComplete = vrai ;
        }
    }
}
```

Commande AT Bluetooth

Mode de commande AT lorsque le module n'est pas connecté.

La commande AT, qui appartient à l'instruction de ligne de caractères, est analysée en fonction de la ligne (c'est-à-dire que la commande AT doit être renvoyée par retour chariot ou \r\n, le nombre hexadécimal est 0D0A). Pour plus de détails sur la commande AT, veuillez vous reporter à Datasheets / BT05-Instruction.pdf. void serialEvent ()

SerialEvent se produit chaque fois qu'une nouvelle donnée arrive dans le RX série matériel. Cette routine est exécutée entre chaque exécution de time loop (), donc l'utilisation de delay inside loop peut retarder la réponse. Plusieurs octets de données peuvent être disponibles.

6.2 Voiture à distance Bluetooth

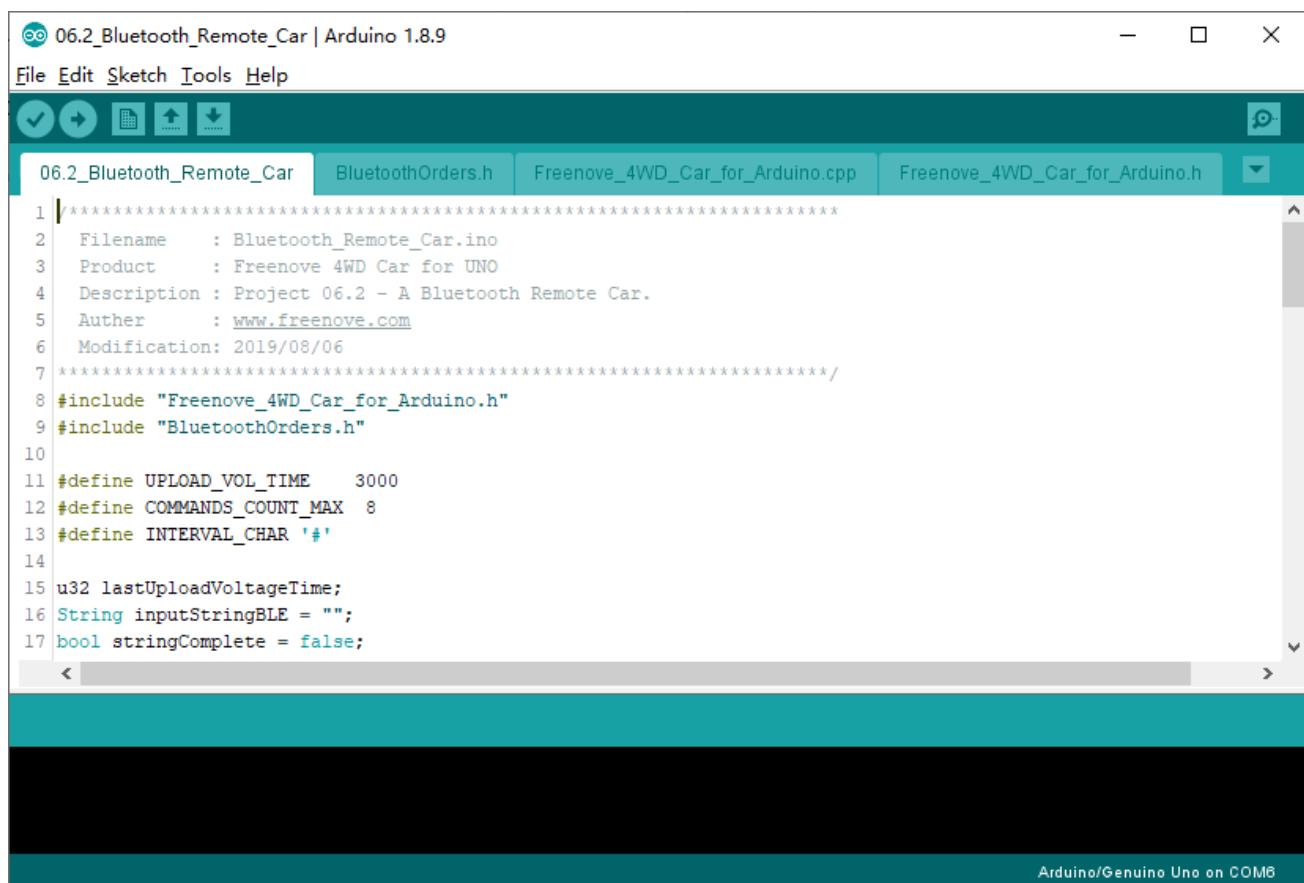
Après avoir appris comment recevoir les données de l'application via Bluetooth et la signification des formats de données, utilisez les données pour laisser la voiture faire quelque chose. Contrôlez le mouvement de la voiture, le buzzer retentit et la LED change le mode d'affichage.

Télécharger le code et exécuter

Vous devez d'abord supprimer le module Bluetooth lorsque vous téléchargez le code.

Et connectez la voiture à l'ordinateur avec un câble USB.

Ensuite, téléchargez le code dans Sketches / 06.2_Bluetooth_Remote_Car.ino.

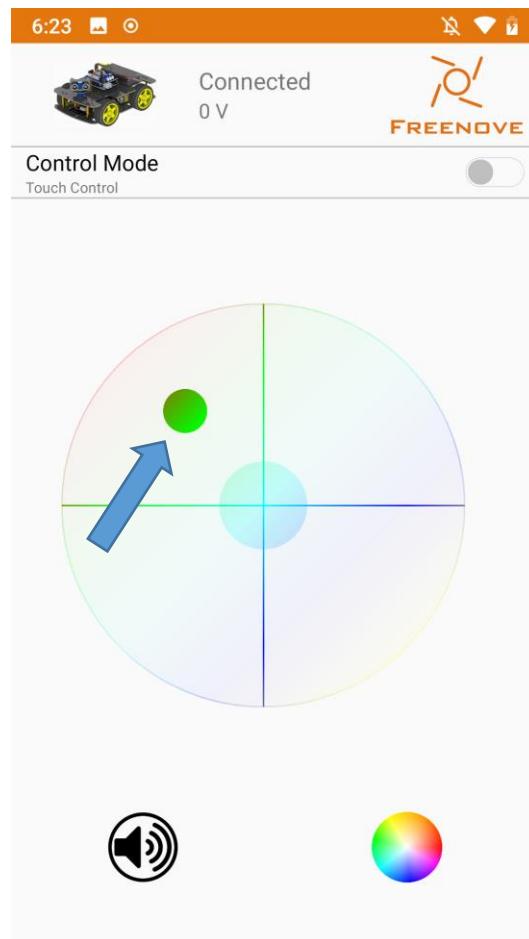


```
06.2_Bluetooth_Remote_Car | Arduino 1.8.9
File Edit Sketch Tools Help
06.2_Bluetooth_Remote_Car BluetoothOrders.h Freenove_4WD_Car_for_Arduino.cpp Freenove_4WD_Car_for_Arduino.h
1 // *****
2 Filename      : Bluetooth_Remote_Car.ino
3 Product       : Freenove 4WD Car for UNO
4 Description   : Project 06.2 - A Bluetooth Remote Car.
5 Author        : www.freenove.com
6 Modification : 2019/08/06
7 *****
8 #include "Freenove_4WD_Car_for_Arduino.h"
9 #include "BluetoothOrders.h"
10
11 #define UPLOAD_VOL_TIME    3000
12 #define COMMANDS_COUNT_MAX 8
13 #define INTERVAL_CHAR '#'
14
15 u32 lastUploadVoltageTime;
16 String inputStringBLE = "";
17 bool stringComplete = false;
```

Arduino/Genuino Uno on COM8

Une fois le code téléchargé, débranchez le câble USB et branchez le module Bluetooth sur la voiture. Allumez l'interrupteur d'alimentation de la voiture.

Selon la méthode précédente, le module Bluetooth de la voiture est connecté via l'application, puis la voiture peut être contrôlée en cliquant ou en faisant glisser le panneau de commande sur l'application.



Code

Le projet comporte 4 libellés, le libellé "BluetoothOrders.h" stocke la communication Bluetooth, et les caractères de commande sont utilisés pour contrôler la voiture, dont seule une partie est utilisée. L'étiquette principale "06.2_Bluetooth_Remote_Car.ino" est le contenu principal de ce projet.

BluetoothOrders.h

```

1 #ifndef _BLUETOOTHORDERS_h
2 #define _BLUETOOTHORDERS_h
3
4 #define ACTION_MOVE 'UNE'
5 #define ACTION_STOP «B»
6 #define ACTION_RGB «C»
7 #define ACTION_BUZZER 'RÉ'
8 #define ACTION_ULTRASONIC «E»
9 #define ACTION_LIGHT_TRACING 'F'
10 #define ACTION_TRACKING 'G'
11 #define ACTION_CAR_MODE «H»

```

```

12 # définir ACTION_GET_VOLTAGE      'JE'
13 # définir ECHO_OK             «J»
14 # définir ACTION_NONE          «K»
15
16 # fin si

```

06.2_Bluetooth_Remote_Car.ino

```

1 # comprendre "Freenove_4WD_Car_for_Arduino.h"
2 # comprendre "BluetoothOrders.h"
3
4 # définir UPLOAD_VOL_TIME      3000
5 # définir COMMANDS_COUNT_MAX 8
6 # définir INTERVAL_CHAR «#»
sept
8 u32 lastUploadVoltageTime ;
9 Chaîne inputStringBLE = "";
dix
10 booléen stringComplete = faux ;
11
12 néant installer () {
13     brochesSetup ();
14     En série . commencer (9600);
15
16 }
17
18 néant boucle () {
19     si ( millis () - lastUploadVoltageTime > UPLOAD_VOL_TIME ) {
20         upLoadVoltageToApp ();
21         lastUploadVoltageTime = millis ();
22     }
23     si ( stringComplete ) {
24         Chaîne inputCommandArray [ COMMANDS_COUNT_MAX ];
25         int paramètres [ COMMANDS_COUNT_MAX ], paramterCount = 0;
26         Chaîne inputStringTemp = inputStringBLE ;
27         pour ( u8 je = 0; je < COMMANDS_COUNT_MAX ; je ++ ) {
28             int indice = inputStringTemp . Indice de ( INTERVAL_CHAR );
29             si ( indice <0) {
30                 Pause ;
31             }
32             paramterCount = je ; //
33             inputCommandArray [ je ] = inputStringTemp . sous-chaîne (0, indice );
34             inputStringTemp = inputStringTemp . sous-chaîne ( indice + 1);
35             paramètres [ je ] = inputCommandArray [ je ]. tolnt ();
36         }
37         stringComplete = faux ;

```

```
38     inputStringBLE = "";
39
40     carboniser commandChar = inputCommandArray [0].caractère (0);
41     commutateur ( commandChar )
42     {
43         Cas ACTION_MOVE :
44             si ( paramterCount == 2) {
45                 motorRun ( paramètres [1], paramètres [2]);
46             }
47             Pause ;
48         Cas ACTION_BUZZER :
49             si ( paramterCount == 1) {
50                 setBuzzer ( paramètres [1]);
51             }
52             Pause ;
53         défaut :
54             Pause ;
55         }
56     }
57 }
58
59 néant upLoadVoltageToApp () {
60     int tension = 0;
61     si ( getBatteryVoltage ()) {
62         tension = Voltage de batterie * 1000;
63     }
64     Chaîne sendString = Chaîne ( ACTION_GET_VOLTAGE ) + Chaîne ( INTERVAL_CHAR ) +
65     Chaîne (( tension )) + Chaîne ( INTERVAL_CHAR );
66     En série . println ( sendString );
67 }
68
69 néant serialEvent () {
70     tandis que ( En série . disponible ()) {
71         carboniser inChar = ( carboniser ) En série . lis ();
72         inputStringBLE += inChar ;
73         si ( inChar == '\n' ) {
74             stringComplete = vrai ;
75         }
76     }
77 }
```

Dans loop (), la valeur de tension est téléchargée dans l'application à intervalles réguliers.

```
    si ( millis () - lastUploadVoltageTime > UPLOAD_VOL_TIME ) {
        upLoadVoltageToApp ();
```

```

        lastUploadVoltageTime = millis ();
    }
}

```

Dans la sous-fonction upLoadVoltageToApp (), lisez d'abord la tension de la batterie, puis convertissez l'unité de tension en mv, et envoyez-le au format "I # xxx #". Utilisez Serial.println () pour envoyer un caractère Newline "\ n", après avoir envoyé les données précédentes,

```

nément upLoadVoltageToApp () {
    int tension = 0;
    si ( getBatteryVoltage () ) {
        tension = Voltage de batterie * 1000;
    }
    Chaîne sendString = Chaîne ( ACTION_GET_VOLTAGE ) + Chaîne ( INTERVAL_CHAR ) +
    Chaîne (( tension )) + Chaîne ( INTERVAL_CHAR );
    En série . println ( sendString );
}

```

Puis dans loop (), si les données requises sont reçues, les données seront analysées en commandes et paramètres et seront sauvegardées dans un tableau.

Supposons que vous recevez une chaîne de commande de "A # 100 # 200 #". Utilisez d'abord la fonction String.indexOf () pour trouver la position du séparateur "#". Et puis utilisez la fonction String.substring () pour diviser la chaîne de commande en "A" et "100 # 200 #", et enregistrez le premier élément "A" qui est divisé dans le tableau "inputCommandArray".

Et puis continuez à répéter cette opération sur la chaîne de fractionnement "100 # 200 #". Vous pouvez diviser l'élément «A» en «100» et «200» respectivement et les enregistrer dans des tableaux. Enfin, convertissez ces éléments en types entiers et enregistrer les dans les paramètres du tableau de paramètres.

```

si ( stringComplete ) {
    Chaîne inputCommandArray [ COMMANDS_COUNT_MAX ];
    int paramètres [ COMMANDS_COUNT_MAX ], paramterCount = 0;
    Chaîne inputStringTemp = inputStringBLE ;
    pour ( u8 je = 0; je < COMMANDS_COUNT_MAX ; je ++ ) {
        int indice = inputStringTemp . Indice de ( INTERVAL_CHAR );
        si ( indice <0 ) {
            Pause ;
        }
        paramterCount = je ; //
        inputCommandArray [ je ] = inputStringTemp . sous-chaîne ( 0, indice );
        inputStringTemp = inputStringTemp . sous-chaîne ( indice + 1 );
        paramètres [ je ] = inputCommandArray [ je ]. tolnt ();
    }
    .....
}

```

finalement , laissez la voiture effectuer différentes actions en fonction des commandes et des paramètres.

```

carboniser commandChar = inputCommandArray [0]. caractère (0);
commutateur ( commandChar )

```

```
{  
    Cas ACTION_MOVE :  
        si ( paramterCount == 2 ) {  
            motorRun ( paramètres [1], paramètres [2]);  
        }  
        Pause ;  
    Cas ACTION_BUZZER :  
        si ( paramterCount == 1 ) {  
            setBuzzer ( paramètres [1]);  
        }  
        Pause ;  
    défaut :  
        Pause ;  
}
```

Indice de()

La description

Localise un caractère ou une chaîne dans une autre chaîne. Par défaut, il recherche à partir du début de la chaîne, mais il peut également commencer à partir d'un index donné, ce qui permet de localiser toutes les instances du caractère ou de la chaîne.

Syntaxe

myString.indexOf (val)

myString.indexOf (val, de)

Paramètres

myString: une variable de type String.

val: la valeur à rechercher. Types de données autorisés: char, String. from: l'index à partir duquel commencer la recherche.

Retour

L'index de val dans la chaîne, ou -1 s'il est introuvable. sous-chaîne ()

La description

Obtenez une sous-chaîne d'une chaîne. L'index de départ est inclusif (le caractère correspondant est inclus dans la sous-chaîne), mais l'index de fin facultatif est exclusif (le caractère correspondant n'est pas inclus dans la sous-chaîne). Si l'index de fin est omis, la sous-chaîne continue jusqu'à la fin de la chaîne.

Syntaxe

myString.substring (à partir de)

myString.substring (de, à)

Paramètres

myString: une variable de type String. from: l'index à partir

duquel démarrer la sous-chaîne.

to (optionnel): l'index pour terminer la sous-chaîne avant.

Retour

La sous-chaîne.

tolnt ()

La description

Convertit une chaîne valide en entier. La chaîne d'entrée doit commencer par un nombre entier. Si la chaîne contient des nombres non entiers, la fonction arrêtera d'effectuer la conversion.

Syntaxe

myString.toInt ()

Paramètres

myString: une variable de type String.

Retour

Si aucune conversion valide n'a pu être effectuée car la chaîne ne commence pas par un nombre entier, un zéro est renvoyé. Type de données: long.

Plus d'informations se référer à : <https://www.arduino.cc/reference/en/language/variables/datatypes/stringobject/>

6.3 Voiture à distance Bluetooth multifonctionnelle

Ce projet ajoute la fonction d'affichage LED et la fonction de commutation de mode d'affichage LED sur la base du projet précédent.

Télécharger le code et exécuter

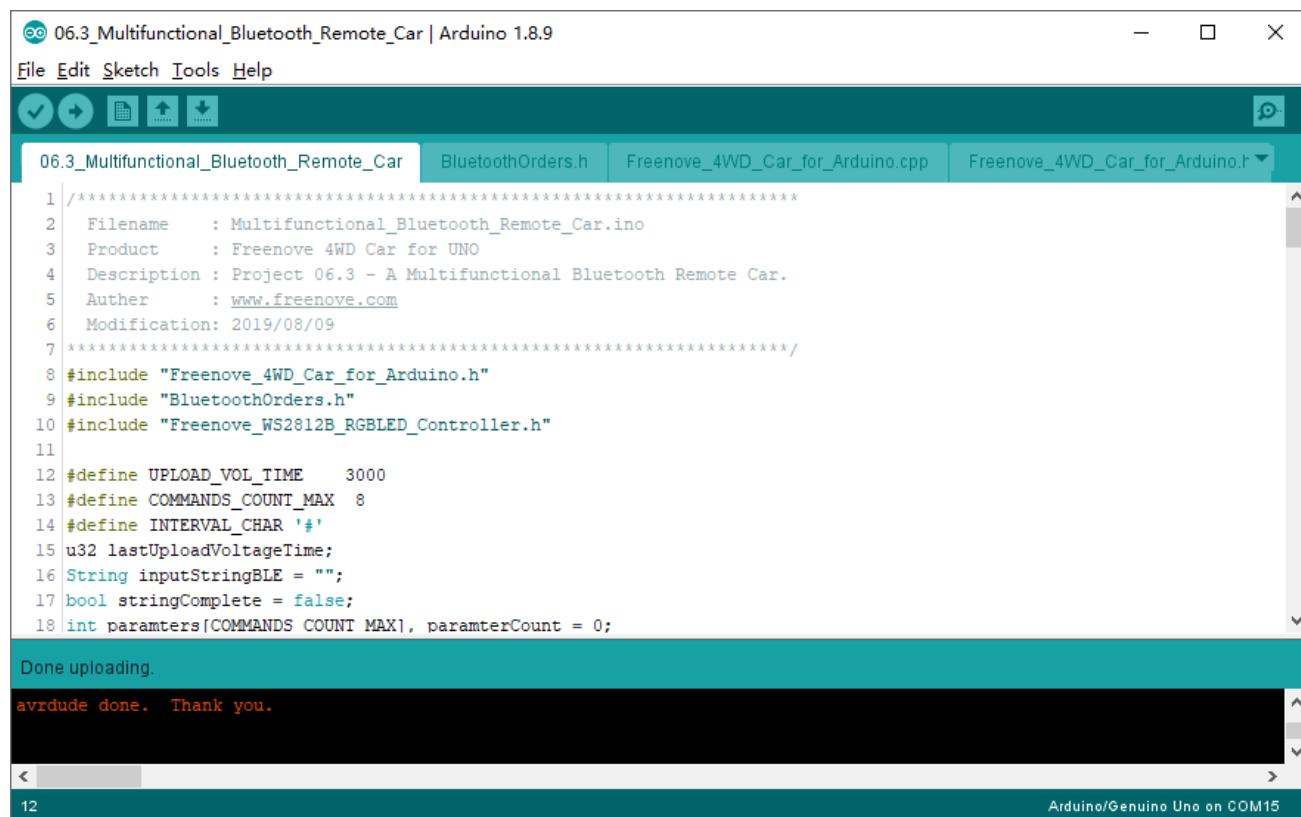
Connectez la voiture à l'ordinateur avec un câble USB.

Vous devez d'abord supprimer le module Bluetooth lorsque vous téléchargez le code.

Ensuite, téléchargez le code dans **Croquis / 06.4_One_Code_Multifunctional_Bluetooth_Remote_Car**.

Après le téléchargement, reconnectez le Bluetooth et **redémarrer** la voiture.

Et vous pouvez télécharger du code dans Sketches / 06.3_Multifunctional_Bluetooth_Remote_Car.ino. Le code est le même. Ne séparez pas les fichiers du dossier.



The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** 06.3_Multifunctional_Bluetooth_Remote_Car | Arduino 1.8.9
- Menu Bar:** File Edit Sketch Tools Help
- Toolbar:** Includes icons for upload, refresh, and other common functions.
- Sketch Navigator:** Shows the files in the sketch: 06.3_Multifunctional_Bluetooth_Remote_Car (selected), BluetoothOrders.h, Freenove_4WD_Car_for_Arduino.cpp, and Freenove_4WD_Car_for_Arduino.h.
- Code Editor:** Displays the C++ code for the project. The code includes header files, defines, and variable declarations. It also includes a section for handling BLE input and a loop that checks for commands.
- Status Bar:** Shows "Done uploading." and "avrduude done. Thank you." followed by a progress bar indicating upload completion.
- Bottom Status:** Shows "12" and "Arduino/Genuino Uno on COM15".

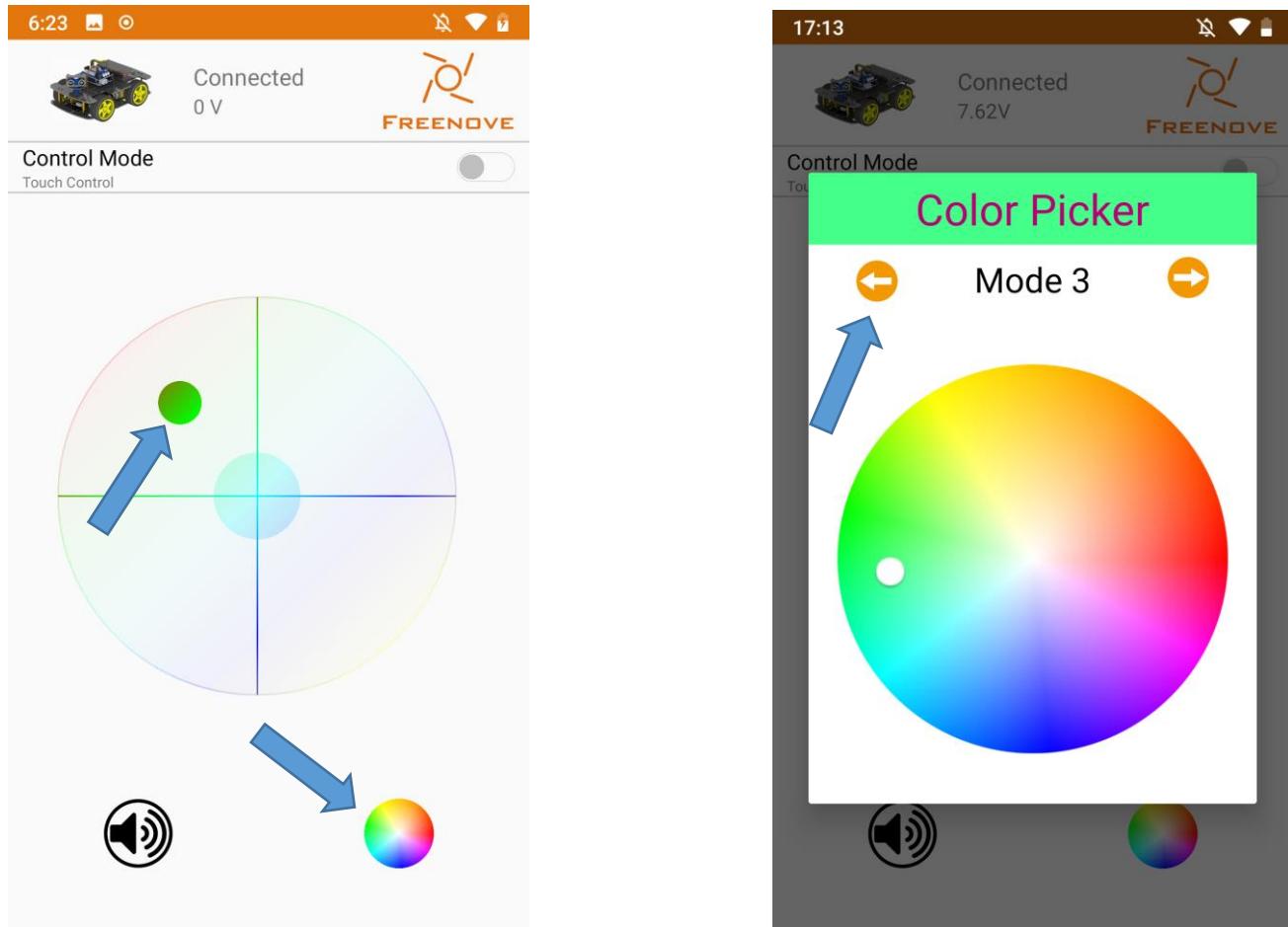
Une fois le code téléchargé, débranchez le câble USB et branchez le module Bluetooth sur la voiture. Allumez l'interrupteur d'alimentation de la voiture.

Selon la méthode précédente, connectez le module Bluetooth de la voiture à l'application, et le panneau de commande de l'application peut être cliqué ou glissé pour contrôler le mouvement de la voiture. Et la palette de couleurs du coin inférieur droit peut être cliquée pour contrôler le mode d'affichage et la couleur de la LED. Parmi eux, le mode 0 est un arc-en-ciel qui coule,

Le mode 1 est une LED à eau courante avec changement de couleur. Le mode 2 est

un clignotement réglable en couleur.

Le mode 3 affiche la couleur sélectionnée du sélecteur de couleur actuel pour toutes les LED.



Code

Le code de ce projet est basé sur le code d'ingénierie précédent, ajoutant un code lié au contrôle des LED. Rien d'autre n'a changé.

06.3_Multifunctional_Bluetooth_Remote_Car

```

1 # comprendre "Freenove_4WD_Car_for_Arduino.h"
2 # comprendre "Automatic_Obstacle_Avoidance_Mode.h"
3 # comprendre "Automatic_Tracking_Line_Mode.h"
4 # comprendre "BluetoothOrders.h"
5 # comprendre "Freenove_WS2812B_RGBLED_Controller.h"
6

```

```
sept # définir UPLOAD_VOL_TIME          3000
8     # définir COMMANDS_COUNT_MAX 8
9     # définir INTERVAL_CHAR «#»
dix   u32 lastUploadVoltageTime;
11    String inputStringBLE = "";
12    booléen stringComplete = faux ;
13    int paramètres [COMMANDS_COUNT_MAX], paramterCount = 0;
14    int bleCarMode = MODE_NONE;
15
16    # définir STRIP_I2C_ADDRESS 0x20
17    # définir STRIP_LEDS_COUNT      dix
18
19    u8 colorPos = 0; u8
20    colorStep = 50;
21    u8 stripDisplayMode = 1; u8
22    currentLedIndex = 0;
23    u16 stripDisplayDelay = 100; u32
24    lastStripUpdateTime = 0;
25    Bande Freenove_WS2812B_Controller (STRIP_I2C_ADDRESS, STRIP_LEDS_COUNT, TYPE_GRB);
26
27    néant installer() {
28        pinsSetup ();
29        Serial.begin (9600);
30        servoSetup ();
31        tandis que (! strip.begin ());
32        strip.setAllLedsColor (0xFF0000);
33    }
34
35    néant boucle() {
36        si (millis () - lastUploadVoltageTime> UPLOAD_VOL_TIME) {
37            upLoadVoltageToApp ();
38            lastUploadVoltageTime = millis ();
39        }
40        si (stringComplete) {
41            String inputCommandArray [COMMANDS_COUNT_MAX];
42
43            String inputStringTemp = inputStringBLE;
44            pour (u8 i = 0; i <COMMANDS_COUNT_MAX; i++) {
45                int index = inputStringTemp.indexOf (INTERVAL_CHAR);
46                si (index <0) {
47                    Pause ;
48                }
49                paramterCount = i; //
50                inputCommandArray [i] = inputStringTemp.substring (0, index);
```

```
51     inputStringTemp = inputStringTemp.substring (index + 1); paramètres [i] =
52         inputCommandArray [i] .toInt ();
53     }
54     stringComplete = faux ;
55     inputStringBLE = "" ;
56
57     carboniser commandChar = inputCommandArray [0] .charAt (0);
58     commutateur (commandChar)
59     {
60         Cas ACTION_MOVE:
61             si (paramterCount == 2) {
62                 motorRun (paramètres [1], paramètres [2]);
63             }
64             Pause ;
65         Cas ACTION_CAR_MODE:
66             si (paramterCount == 1) {
67                 bleCarMode = paramètres [1];
68                 commutateur (bleCarMode)
69                 {
70                     Cas MODE_NONE: Cas MODE_GRAVITY:
71                         resetCarAction ();
72                         writeServo (OA_SERVO_CENTER);
73                         Pause ;
74                     Cas MODE_ULTRASONIC:
75                         oa_CalculateVoltageCompensation ();
76                         Pause ;
77                     Cas MODE_TRACKING:
78                         tk_CalculateVoltageCompensation ();
79                         Pause ;
80                     défaut :
81                         Pause ;
82                 }
83             }
84             Pause ;
85         Cas ACTION_BUZZER:
86             si (paramterCount == 1) {
87                 setBuzzer (paramètres [1]);
88             }
89             Pause ;
90         Cas ACTION_RGB:
91             si (paramterCount == 4) {
92                 stripDisplayMode = paramètres [1];
93                 commutateur (stripDisplayMode)
94                 {
```

```
95          Cas 0:  
96              colorStep = 5;  
97              stripDisplayDelay = 100;  
98              Pause ;  
99          Cas 1:  
100         colorStep = 50;  
101         stripDisplayDelay = 50;  
102         Pause ;  
103         Cas 2:  
104         colorStep = 5;  
105         stripDisplayDelay = 500;  
106         Pause ;  
107         Cas 3:  
108         Pause ;  
109         défaut :  
110         Pause ;  
111     }  
112 }  
113 Pause ;  
114 défaut :  
115 Pause ;  
116 }  
117 }  
118 commutateur (bleCarMode)  
119 {  
120     Cas MODE_NONE: Cas MODE_GRAVITY:  
121         Pause ;  
122     Cas MODE_ULTRASONIC:  
123         upLoadSonarValueToApp ();  
124         updateAutomaticObstacleAvoidance ();  
125         Pause ;  
126     Cas MODE_TRACKING:  
127         updateAutomaticTrackingLine ();  
128         Pause ;  
129         défaut :  
130         Pause ;  
131     }  
132     statique u8 lastColor [3];  
133     commutateur (stripDisplayMode)  
134 {  
135     Cas 0:  
136         si (millis () - lastStripUpdateTime > stripDisplayDelay) {  
137             pour (int i = 0; i < STRIP_LEDS_COUNT; i++) {  
138                 pour (int j = 0; j < 3; j++) {  
139                     lastColor[j] = (lastColor[j] + colorStep) % 256;
```

```
139         strip.setLedColorData (i, strip.Wheel (colorPos + i * 25));
140     }
141     strip.show ();
142     colorPos += colorStep;
143     lastStripUpdateTime = millis ();
144 }
145 Pause ;
146 Cas 1:
147 si (millis () - lastStripUpdateTime > stripDisplayDelay) {
148
149     strip.setLedColor (currentLedIndex, strip.Wheel (colorPos)); currentLedIndex++;
150
151     si (currentLedIndex == STRIP_LEDS_COUNT) {
152
153         currentLedIndex = 0;
154         colorPos += colorStep; //
155     }
156     lastStripUpdateTime = millis ();
157 }
158 Pause ;
159 Cas 2:
160 colorPos = colorStep;
161 si (millis () - lastStripUpdateTime > stripDisplayDelay) {
162
163     booléen statique ledState = vrai ;
164     si (ledState)
165     {
166         strip.setAllLedsColor (paramètres [2], paramètres [3], paramètres [4]);
167     }
168     autre
169     {
170         strip.setAllLedsColor (0x00);
171     }
172     ledState =! ledState;
173     lastStripUpdateTime = millis ();
174 }
175 Pause ;
176 Cas 3:
177 si (lastColor [0] != paramètres [2] || lastColor [1] != paramètres [3] || lastColor [2] != paramètres [4])
178
179 {
180     strip.setAllLedsColor (paramètres [2], paramètres [3], paramètres [4]); lastColor [0] =
181     paramètres [2];
182     lastColor [1] = paramètres [3];
```

```

183     lastColor [2] = paramètres [4];
184 }
185 Pause ;
186 défaut :
187     Pause ;
188 }
189 }
190
191 néant upLoadVoltageToApp () {
192     int tension = 0;
193     si (getBatteryVoltage ()) {
194         tension = batteryVoltage * 1000;
195     }
196     String sendString = String (ACTION_GET_VOLTAGE) + String (INTERVAL_CHAR) + String ((voltage)) + String (INTERVAL_CHAR);
197
198     Serial.println (sendString);
199 }
200
201 externe int distance [3];
202 néant upLoadSonarValueToApp () {
203     String sendString = String (ACTION_ULTRASONIC) + String (INTERVAL_CHAR) + String ((distance [1]))
204     + String (INTERVAL_CHAR);
205     Serial.println (sendString);
206 }
207
208 néant serialEvent () {
209     tandis que (Série disponible ()) {
210         carboniser inChar = ( carboniser ) Serial.read ();
211         inputStringBLE += inChar;
212         si (inChar == '\n') {stringComplete =
213             vrai ;
214         }
215     }
216 }
```

Il y a trois modes d'action. MODE_GAVITY, MODE_ULTRASONIC et MODE_TRACKING.

```

Cas ACTION_CAR_MODE:
    si (paramterCount == 1) {
        bleCarMode = paramètres [1];
        commutateur (bleCarMode)
    }
    Cas MODE_NONE: Cas MODE_GRAVITY:
        resetCarAction ();
        writeServo (OA_SERVO_CENTER);
```

```

        Pause ;

Cas MODE_ULTRASONIC:
    oa_CalculateVoltageCompensation ();
    Pause ;

Cas MODE_TRACKING:
    tk_CalculateVoltageCompensation ();
    Pause ;
défaut :
    Pause ;
}
}

Pause ;

```

Dans l'application, quatre modes d'affichage LED peuvent être commutés et différentes plages de changement de couleur et périodes de changement de couleur sont définies en fonction de différents modes. Autrement dit, les valeurs des variables colorStep et

stripDisplayDelay.

```

commutateur (commandChar)
{
    Cas ACTION_MOVE:
        si (paramterCount == 2) {
            motorRun (paramètres [1], paramètres [2]);
        }
        Pause ;

    Cas ACTION_CAR_MODE:
        si (paramterCount == 1) {
            bleCarMode = paramètres [1];
            commutateur (bleCarMode)
        }

        Cas MODE_NONE: Cas MODE_GRAVITY:
            resetCarAction ();
            writeServo (OA_SERVO_CENTER);
            Pause ;

        Cas MODE_ULTRASONIC:
            oa_CalculateVoltageCompensation ();
            Pause ;

        Cas MODE_TRACKING:
            tk_CalculateVoltageCompensation ();
            Pause ;
        défaut :
            Pause ;
    }
}

Pause ;

Cas ACTION_BUZZER:

```

```
    si (paramterCount == 1) {
        setBuzzer (paramètres [1]);
    }
    Pause ;
    Cas ACTION_RGB:
    si (paramterCount == 4) {
        stripDisplayMode = paramètres [1];
        commutateur (stripDisplayMode)
    }
    Cas 0:
    colorStep = 5;
    stripDisplayDelay = 100;
    Pause ;
    Cas 1:
    colorStep = 50;
    stripDisplayDelay = 50;
    Pause ;
    Cas 2:
    colorStep = 5;
    stripDisplayDelay = 500;
    Pause ;
    Cas 3:
    Pause ;
    défaut :
    Pause ;
}
}
Pause ;
défaut :
Pause ;
}
```

En boucle (), la LED est affichée en fonction du mode reçu et des paramètres.

```
statique u8 lastColor [3];
commutateur (stripDisplayMode)
{
    Cas 0:
    si ( millis () - lastStripUpdateTime > stripDisplayDelay ) {

        pour ( int je = 0; je < STRIP_LEDS_COUNT ; je ++ ) {bande. setLedColorData ( je , bande. Roue (colorPos
            + je * 25));
        }
        bande. montrer ();
    }
}
```

```

        colorPos += colorStep;
        lastStripUpdateTime = millis ();
    }
    Pause ;
    Cas 1:
    si ( millis () - lastStripUpdateTime > stripDisplayDelay) {

        bande. setLedColor (currentLedIndex, strip. Roue (colorPos)); currentLedIndex++;

        si (currentLedIndex == STRIP_LEDS_COUNT ) {

            currentLedIndex = 0;
            colorPos += colorStep; //

        }
        lastStripUpdateTime = millis ();
    }
    Pause ;
    Cas 2:
    colorPos = colorStep;
    si ( millis () - lastStripUpdateTime > stripDisplayDelay) {

        statique booléen ledState = vrai ;
        si (ledState)
        {
            bande. setAllLedsColor (paramètres [2], paramètres [3], paramètres [4]);
        }
        autre
        {
            bande. setAllLedsColor (0x00);
        }
        ledState =! ledState;
        lastStripUpdateTime = millis ();
    }
    Pause ;
    Cas 3:
    si (lastColor [0]! = paramètres [2] || lastColor [1]! = paramètres [3] || lastColor [2]! = paramètres [4])

    {
        bande. setAllLedsColor (paramètres [2], paramètres [3], paramètres [4]); lastColor [0] =
        paramètres [2];
        lastColor [1] = paramètres [3];
        lastColor [2] = paramètres [4];
    }
    Pause ;

```

```
    défaut :  
        Pause ;  
    }  
}
```

Quelle est la prochaine?

MERCI d'avoir participé à cette expérience d'apprentissage!

Nous avons atteint la fin de ce didacticiel. Si vous trouvez des erreurs, des omissions ou si vous avez des suggestions et / ou des questions sur le didacticiel ou le contenu des composants de ce kit, n'hésitez pas à nous contacter:

support@freenove.com

Nous ferons tout notre possible pour apporter des modifications et corriger les erreurs dans les meilleurs délais et publier une version révisée.

Si vous souhaitez en savoir plus sur Arduino, Raspberry Pi, Smart Cars, Robotics et autres produits intéressants en science et technologie, veuillez continuer à visiter notre site Web. Nous continuerons de lancer des produits amusants, rentables, innovants et passionnnants.

<http://www.freenove.com/>

Merci encore d'avoir choisi les produits Freenove.