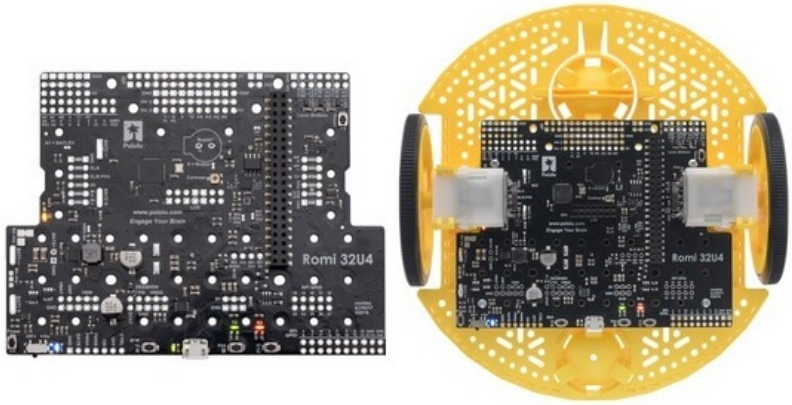


Carte de contrôle Romi 32U4 - guide utilisateur

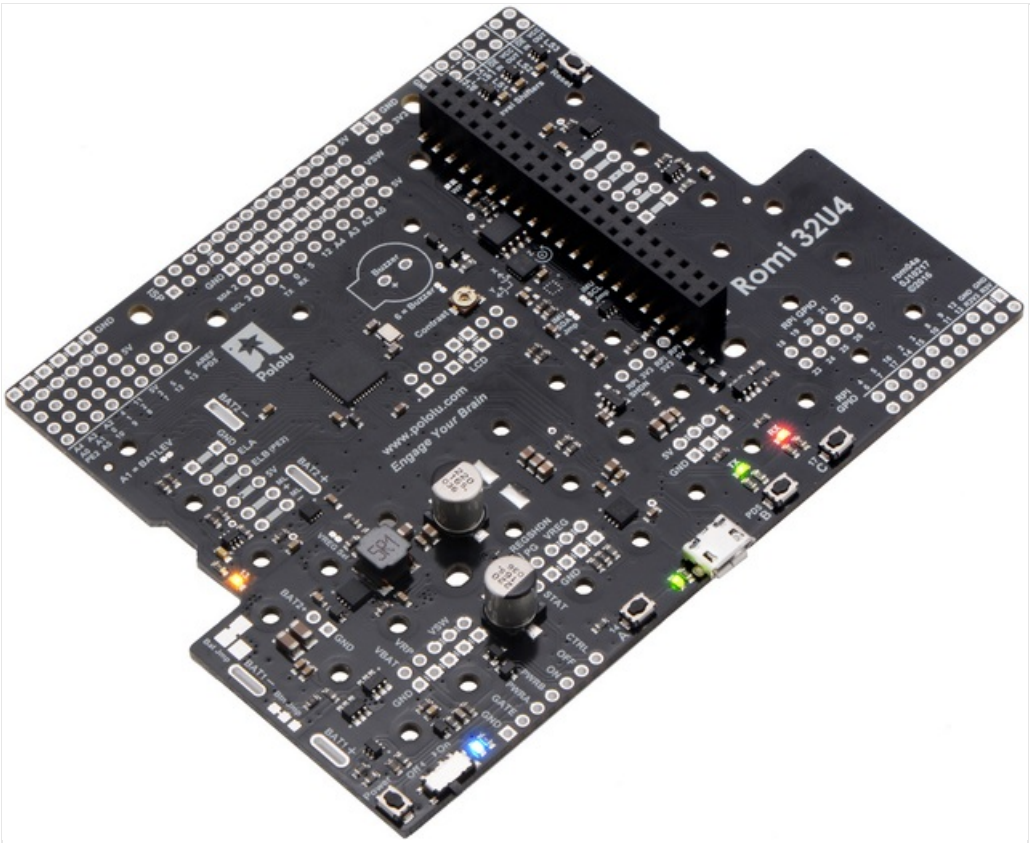
Apprenez comment utiliser et exploiter la carte contrôleur conçue pour la plateforme Robotique Romi

(version 0.1)

Traduit par MicroControleur Hobby (shop.mchobby.be)
Compilé depuis la traduction maintenue sur https://wiki.mchobby.be/index.php?title=Pololu-Romi-32U4
Les hyperliens sont disponibles sur la version en ligne du document.
Translated by MicroControleur Hobby (shop.mchobby.be)
Compiled from online translation available at https://wiki.mchobby.be/index.php?title=Pololu-Romi-32U4
Hyperlinks are available on the online version of this document.

Introduction

La carte contrôleur Romi 32U4 (Lien Pololu https://www.pololu.com/category/149/a-star-programmable-controllers) est conçue pour être montée sur le châssis Romi pour créer une plateforme robotique qui peut facilement être programmée et personnalisée.



Carte de contrôle Romi 32U4

Cliquer l'image pour l'agrandir

Comme pour le contrôleur programmable A-Star 32U4 de Pololu, la carte de contrôle Romi 32U4 est réalisé autour d'un AVR Atmel ATmega32U4 disposant déjà de fonctionnalités USB. Il est par ailleurs préchargé avec un bootloader compatible Arduino. La carte de contrôle dispose de deux contrôleurs moteurs (des ponts-H) et prévoit des connexions pour y connecter une paire d'encodeur Romi en kit https://shop.mchobby.be/product.php?id_product=1452 lien pololu https://www.pololu.com/product/3542 (disponible séparément) pour permettre un contrôle en boucle fermée. La carte inclus également un puissant régulateur de tension 5 V de type step-down capable de maintenir un courant de sortie de 2A, accompagné d'un circuit de coupure et distribution d'alimentation. Un accéléromètre 3-axes et gyroscope permet au Robot Romi 32U4 de faire des mesures inertielles, estimer son orientation et détecter les forces appliquées.

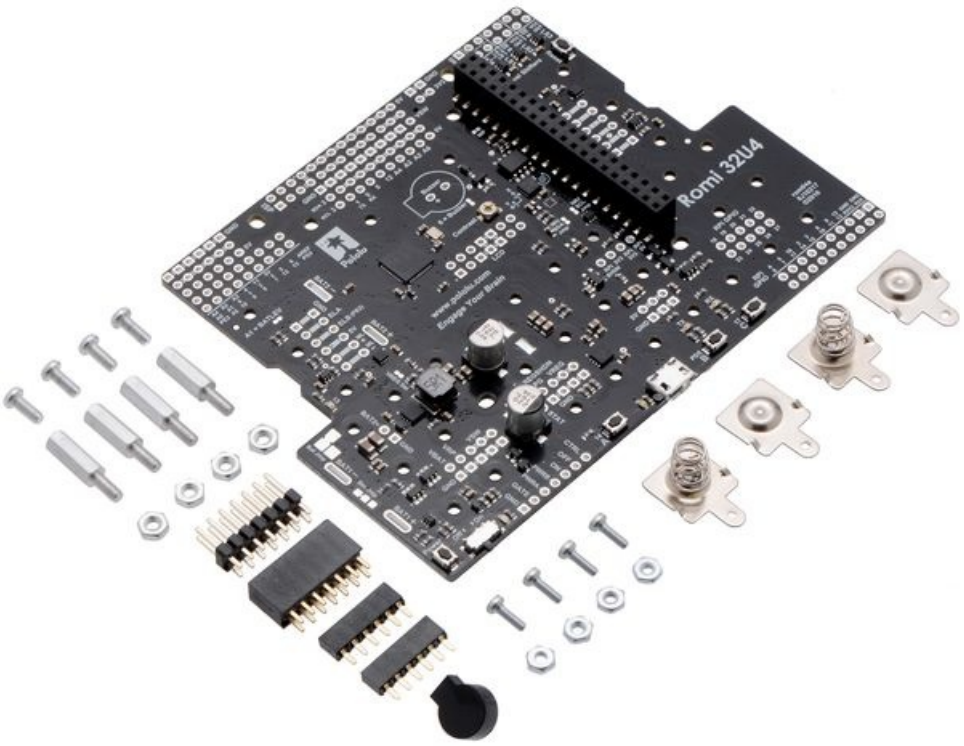
Les trois boutons poussoirs permettent de réaliser une interface utilisateur tandis que les LEDs indicatrices, buzzer et connecteur (pour l'afficheur LCD) permettent d'offrir un retour d'information.

La carte de contrôle Romi 32U4 peut être utilisé pour réaliser une solution de contrôle autonome **OU** comme base avec un Raspberry-Pi pour réaliser un puissant contrôleur robotique. Les connecteurs de la carte et trous de montage permettent de brancher directement un Raspberry Pi comptaible https://shop.mchobby.be/category.php?id_category=30 (Modèle B+ ou plus récent, incluant le Pi 3 modèle B et modèle A+). Les convertisseurs de niveau logique (*level shifters* en anglais) permettent de mettre facilement en place une communication I^2C ou d'interfacer d'autres signaux entre les deux contrôleurs. La carte de contrôle Romi fournit automatiquement la tension d'alimentation 5 V au Raspberry-Pi branché sur la carte. Dans cette configuration, le Raspberry Pi peut prendre en charge le contrôle de haut niveau du robot tandis qu'il s'appuie sur la carte de contrôle Romi 32U4 pour les tâches de bas niveau (comme alimenter les moteurs, lire les encodeurs, l'interface avec des capteurs analogiques ou les capteurs ayant un protocole très sensible d'un point de vue temporel).

Les lignes d'entrée / sortie de l'ATmega32U4 et du Raspberry Pi sont disponibles en breakout sur des connecteurs 2.54mm à l'avant et arrière de la carte de contrôle. Les lignes d'alimentation sont également accessibles à l'avant et à l'arrière, ce qui permet de brancher facilement des capteurs sur la plateforme.

Un greffons logiciel est disponible pour programmer facilement un robot Romi 32U4 depuis l'environnement Arduino, Pololu propose des bibliothèques Arduino et croquis d'exemples pour vous aider dans vos premiers pas. Un câble USB A vers microB (non inclus) est nécessaire pour programmer la carte.

Composant inclus



Eléments inclus

Cliquer l'image pour l'agrandir

Les éléments suivants sont inclus avec la carte de contrôle Romi 32U4:

- Deux connecteurs femelles (bas profile) pour les moteurs et encodeurs
- buzzer lien pololu https://www.pololu.com/product/1484
- Connecteurs femelles 2x7 broches lien pololu https://www.pololu.com/product/1027 et connecteurs mâle lien pololu https://www.pololu.com/product/966 pour l'afficheur LCD
- Connexion pour les piles
- 4 vis et écrou 3/16" #2-56
- 4 entretoises M2.5 lien pololu https://www.pololu.com/product/1952 (de 11 mm de long), vis et écrous pour monter un Raspberry Pi

Le LCD et le Raspberry-Pi ne sont pas inclus avec la carte contrôleur Romi 32U4.

Eléments nécessaires

Ces éléments complémentaires sont également nécessaires pour utiliser et assembler la carte contrôleur Romi 32U4:

Accessoires nécessaires

- Un Kit châssis Romi https://shop.mchobby.be/product.php?id_product=1448 lien pololu https://www.pololu.com/product/203 (cela inclus les moteurs, roues et bille libre et contact de piles)
- Une paire d'encodeur Romi (kit) https://shop.mchobby.be/product.php?id_product=1452 lien pololu https://www.pololu.com/product/3542
- 6x piles AA. Le châssis Romi et la carte de contrôle fonctionne avec des piles Alcalines et piles NiMH. Nous recommandons néanmoins des piles rechargeables NiMH.

Outils de montage

- Un fer à souder et de la soudure
- Un petit tournevis à croix (dit Phillips)
- USB A to Micro-B cable https://shop.mchobby.be/product.php?id_product=68 lien pololu https://www.pololu.com/product/2072 to connect the board to your computer for programming and debugging

Outils complémentaires

- Un petit tournevis plat 2 mm pour ajuster le contrast de l'écran LCD.
- Une pince à bec https://shop.mchobby.be/product.php?id_product=705 lien pololu https://www.pololu.com/product/150
- Une pince à dénuder multi-fonctionnelle https://shop.mchobby.be/product.php?id_product=162 lien pololu https://www.pololu.com/product/1923 , pour ajouter des fils pour périphériques
- De la toile isolante ou petite pince (pour maintenir les pièces ensemble au moment de la soudure)

Accessoires optionnels

Vous pourriez également opter pour l'un des accessoires ci-dessous pour agrémenter votre Robot Romi 32U4:

- Un afficheur LCD 8×2 caractères lien pololu https://www.pololu.com/product/356
- Un compatible Raspberry Pi https://shop.mchobby.be/category.php?id_category=30 (Modèle B+ ou plus récent, incluant le Pi 3 modèle B et modèle A+)
- Capteurs / senseurs, tels que des capteurs optiques ou ultrasoniques
- Connecteurs et fils de prototypage pour connecter des composants et capteurs additionnels
- Chargeur de pile (si vous utilisez des piles rechargeables); puisque le Romi utilise des piles AA, Pololu recommande un chargeur de piles AA standard (de ceux dans lesquels il est possible de glisser les piles individuellement) que l'on trouve très facilement. Popolu propose également le chargeur iMAX-B6AC V2 (*balance charger/discharger*) qui peut également être utilisé pour cela.

Systèmes d'exploitations supportés

La carte de contrôle Romi 32U4 peut être programmée sur n'importe quel système d'exploitation qui supporte l'environnement Arduino.

Cela inclus:

- Microsoft Windows 10, 8.1, 8, 7, Vista, XP (with Service Pack 3),
- Linux,
- macOS.

A Propos de MCHobby
Crédit de traduction

Sommaire

A propose de .: MC Hobby :.

MCHobby investit du temps et de l'argent dans la réalisation de traduction et/ou documentation. C'est un travail long et fastidieux réalisé dans l'esprit Open-Source... donc gratuit et librement accessible.
SI vous aimez nos traductions et documentations ALORS aidez nous à en produire plus en achetant vos produits chez MCHobby https://shop.mchobby.be .

MC Hobby SPRL, basé en Belgique, est le traducteur de ce manuel.

MC Hobby cherche à promouvoir, en français, la plateforme Open-Source Arduino, ses extensions et exemples de programmation afin de la rendre accessible au plus grand nombre.

Licence

Crédit de traduction

Toute référence, mention ou extrait de cette traduction doit être explicitement accompagné du texte de crédit de traduction.

Ce texte est repris en "pied de document" sur toutes les pages/documents, merci d'en prendre connaissance.

Limite de traduction

Anglicisme

Du fait que de nombreux anglicismes soient utilisés au jour le jour dans les milieux techniques, je me suis permis d'en préserver quelques-uns qui me semblaient avoir plus de sens dans la langue de Shakespeare que traduit dans la langue de Voltaire. Par ailleurs, ces anglicismes pourraient se révéler fort utiles lors de vos prochaines recherches sur Internet.

Ainsi, vous retrouverez les termes suivants :

- Pin : fait référence à une « broche » de raccordement d'un composant (Anglicisme fort répandu).
- Datasheet : fait référence à la « fiche technique » d'un composant.
- Pin Header : malheureusement intraduisible sans en perdre le sens mais vous aurez vite identifié les « Pinheaders ». La traduction courant est *connecteur*, ce terme restant beaucoup trop vague.

• LED : ce sont les diodes électroluminescentes (aussi appelées DEL). Cependant, le terme LED est si répandu qu'il a été préservé. • RGB : Fait référence aux 3 couleurs fondamentales Rouge Vert Bleu (Red Green Blue). Si le terme RVB est d'usage fréquent en Belgique et en France, l'acronyme RGB reste encore le plus répandu.

Code source

Nous avons traduit les commentaires dans les programmes afin de les rendre plus intelligibles.

Par contre, nous n'avons pas modifié les codes sources (nom des variables, ...) qui, eux, restent ceux fournit par le concepteur et le fabricant du kit.

Autorisations

Le présent manuel a été traduit avec l'autorisation de Pololu (www.pololu.com).

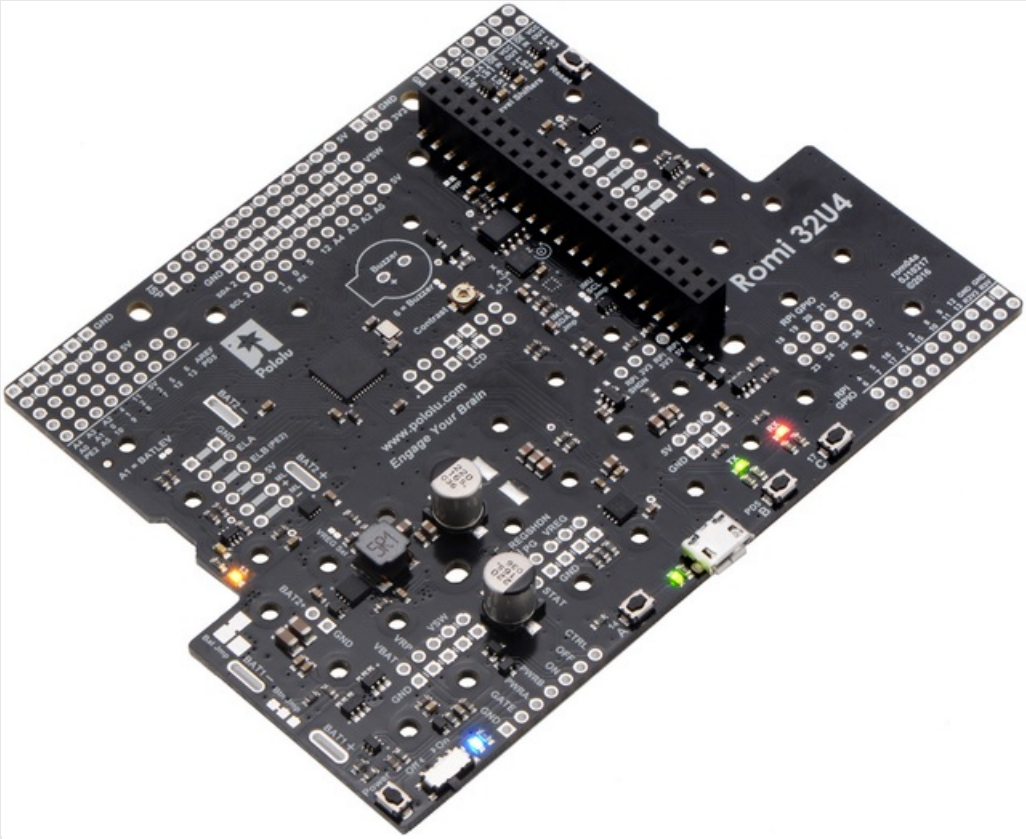
Signaler une erreur

Malgré tout le soin apporté à la réalisation de cette traduction, il n'est pas impossible qu'une erreur se soit malgré tout glissée dans ce document. N'hésitez pas à nous envoyez un e-mail si vous en constatiez une. Pour adresser vos remarques et commentaires relatifs à la traduction, ou pour demander une traduction complémentaire, contactez nous par e-mail à

- support (arobase) mchobby.be.

N'oubliez pas de mentionner le manuel/page/lien en mentionnant l'erreur ;-)

Contacter Pololu



Carte de contrôle Romi 32U4

Cliquer l'image pour l'agrandir

Pololu serait ravit d'être informé de vos projects et retour d'expérience avec le châssis Romi. Vous pouvez contacter Pololu https://www.pololu.com/contact directement ou écrire un message sur les Forums Pololu https://forum.pololu.com/ . N'hésitez pas à communiquer ce qui est bien, ce qui peut être améliorer, les amélioration souhaitées ou tout ce que vous désireriez partager!

Microcontrôleur

Comme les microcontrôleurs A-Star 32U4 de Pololu (lien Pololu https://www.pololu.com/category/149/a-star-programmable-controllers), la carte de contrôle Romi 32U4 met en oeuvre un microcontrôleur AVR ATmega32U4 d'Atmel disposant d'un support USB natif, d'un oscillateur cristal de précision à 16 MHz. C'est le même microcontrôleur et même fréquence d'un Arduino Leonardo lien pololu https://www.pololu.com/product/2192 et Arduino Micro https://shop.mchobby.be/product.php?id_product=451 lien pololu https://www.pololu.com/product/2188 .

La carte contrôleur inclus un connecteur USB micro-B qui permet de connecter la carte sur un ordinateur à l'aide d'un câble USB A vers micro-B https://shop.mchobby.be/product.php?id_product=68 lien pololu https://www.pololu.com/product/2072 (non inclus). La connexion USB peut être utilisé pour transmettre et recevoir des informations vers un ordinateur et pour programmer la carte via USB. La connexion USB peut également alimenter le microcontrôleur ainsi que la plupart des composants de la carte (à l'exception de la puissance moteur); Point décrit plus en détail dans la section "alimentation".

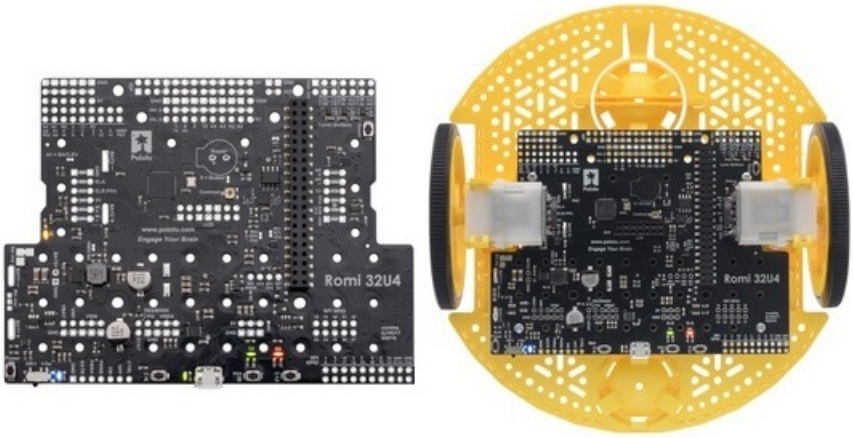
La carte de contrôle ATmega32U4 est livrée avec un bootloader USB A-Star 32U4 Arduino compatible, ce qui permet de programmer facilement la carte avec Arduino IDE. Pour plus d'information sur la programmation de la carte contrôleur Romi 32U4, voyez la section "Programmer la carte Romi 32U4".

La carte dispose également d'un connecteur ISP 6 broches qui permet de programmer le microcontrôleur à l'aide d'un programmeur externe, comme le programmeur AVR USB de Pololu lien pololu https://www.pololu.com/product/3172 . La broche 1 du connecteur est repérée à l'aide d'un petit point blanc et à une forme octogonale.

Interface utilisateur

Sommaire

LEDs



La carte de contrôle Romi 32U4 dispose de 5 LEDs indicatrices.

- Une LEDs **jaune** qui est connectée sur la broche digital 13 d'Arduino, ou PC7. Vous pouvez placer cette broche au niveau dans le programme utilisateur pour alumer la LED. Le bootloader 32U4 de l'A-Star fait pulser cette LED pendant qu'il attend le chargement d'un nouveau croquis.
- Une LED utilisateur **verte** est connectée sur la broche Arduino 30, ou PD5, et est allumée lorque la broche est placée au niveau bas. Lorsque la carte exécute le bootloader 32U4 de l'A-Star (ou un programme compilé dans l'environnement Arduino), cette LED flashera lorsqu'il y a des transmission de données via la connexion USB.
- Une LED utilisateur **rouge** est connectée sur la broche Arduino #17, ou PB0, et est allumée lorsque la pin est au niveau bas. Lorsque la carte exécure le bootloader 32U4 de l'A-Start (ou un programme compilé dans l'environnement Arduino), elle clignotera lorsque des données sont reçue via la connexion USB.

La bibliothèque Romi32U4 contient des fonction qui facilite le contrôle de des 3 LEDs utilisateurs. Les lignes de contrôle des LEDs utilisateurs sont également des lignes de données de l'écran LCD. Par conséquent, elle scintillent pendant la mise-à-jour du LCD. Les LED verte et rouge partagent également les lignes E/S des boutons poussoirs (voir plus bas).

- Une LED d'alimentation **bleue** à côté de l'interrupteur d'alimentation indique que le contrôleur reçoit une alimentation depuis les piles du Romi (Le circuit d'alimentation doit être sur la position marche).
- Une led d'alimentation **verte** près du connecteur USB (sur le bas de la carte) indique qu'une tension est présente sur le bus USB (VBUS).

Boutons poussoirs

La carte de contrôle Romi 32U4 dispose de 5 boutons poussoir:

- Un **bouton d'alimentation** (*Power*) sur le coin arrière gauche,
- Un '*bouton Reset* sur le côté avant droit,
- Et **trois boutons utilisateurs** placé sur le bord arrière.

Les boutons utilisateurs, libellés A, B et C sont respectivement raccordés aux broches Arduino 14 (PB3), 30 (PD5) et 17 (PB0). Presser un de ces bouton place le niveau logique de la broche au niveau bas(à la masse via une résistance).

Ces trois lignes E/S des boutons sont également utilisé pour d'autres fonctionnalités: la broche 14 est MISO sur l'interface SPI, la broche 30 et 17 contrôle les LEDs utilisateurs verte et rouge. Et ces trois lignes d'E/S sont également les lignes de données de l'afficheur LCD. Cependant ces autres usages nécessite que les broches soient pilotées par l'AVR (ou le périphérique SPI esclave dans le cadre de MISO), les résistances présentes dans les circuits des bouton assurent que la carte de contrôle Romi 32U4 ne sera pas endommagée même si le bouton correspondant

est pressé en même temps que le pilotage par l'AVR, même si la communication SPI ou LCD sera corrompue. Les fonctions de la bibliothèque Romi32U4 prend en charge la configuration des broches, lecture et déparasitage des boutons et restauration des broches dans leurs états initials.

Afficheur LCD

La carte de contrôle Romi 32U4 dispose d'un emplacement 2x7 broches pour placer un connecteur pour l'écran LCD 8×2 caractères lien pololu https://www.pololu.com/product/356 (ou tout autre LCD utilisant l'interface parallèle HD44780 https://www.pololu.com/file/0J71/DMC50448N-AAE-AD.pdf (109k pdf)). Vous pouvez ajuster le contraste du LCD avec le potentiomètre en haut à droite du connecteur LCD. Pololu recommande d'utiliser un tournevis pour ajuster le contraste.

La bibliothèque Romi32U4 propose des fonctions qui affiche des données sur le LCD connecté. La bibliothèque est conçue pour gérer les fonctions alternatives des lignes de données même lorsque les broches sont requises pour le passage de commandes LCD, après quoi, les broches seront restaurées dans leur précédent état. Cela permet d'utiliser les lignes de données LCD pour d'autres fonctions (tel qu'une entrée bouton poussoir et pilotage de LEDs).

A noter que la carte de contrôle n'est pas conçue pour recevoir simultanément un afficheur LCD et un Raspberry-Pi branché sur le port 40 broches. Cependant, avoir un connecteur LCD soudé sur la carte ne devrait pas interférer avec le placement d'un Raspberry Pi.

Buzzer

Le buzzer lien pololu https://www.pololu.com/product/1484 inclus sur la carte de contrôle Romi 32U4 peut être soudé sur l'emplacement traversant prévu à cet effet. Cela permet de générer des sons simples et de la musique. Le buzzer est connecté sur la broche digitale 6 (qui est aussi utilisée comme OC4D, une sortie PWM matérielle de l'AVR exploitant le Timer4 10-bits). Si la broche passe alternativement d'un niveau haut à un niveau bas à une fréquence donnée alors le buzzer produira un son à cette fréquence. Il est possible de jouer des notes et de la musique en utilisant des fonction de la bibliothèque Romi32U4. Si vous désirez utiliser la broche 6 pour une autre fonctionnalité (autre que la fonction de buzze), il est possible de déconnecter le buzzer en sectionnant cavalier CMS/SMD à côté du buzzer.

Contrôleur moteur et encodeurs

Contrôleur moteurs

La carte contrôleur ROMI 32U4 est équipé de deux contrôleur moteur DRV8838 de Texas Instruments qui sont utilisés pour la puissance délivrée aux deux mini moteurs plastique avec boîte de vitesse lien pololu https://www.pololu.com/product/1520 . Quatres broches de l'Arduino sont utilisées pour piloter les contrôleurs:

- **Broche digital 15** (ou PB1) contrôle le **sens de rotation du moteur droit** (LOW drives the motor forward, HIGH drives it in reverse).
- **Broche digital 16** (ou PB2) contrôle le **sens de rotation du moteur gauche**.
- **Broche digital 9** (ou PB5) contrôle la **vitesse du moteur droit** avec du PWM (*Pulse Width Modulation*, modulation de longueur d'impulsion) généré par le Timer1 de l'ATmega32U4.
- **Broche digital 10** (ou PB6) contrôle la **vitesse du moteur gauche** avec du PWM.

Pour plus d'information sur les contrôleurs moteurs, voyez la fiche technique du DRV8838 https://www.pololu.com/file/0J806/drv8838.pdf (*1 Mio, pdf*). Popolu propose également une carte contrôleur moteur dit *carrier board* lien pololu https://www.pololu.com/product/2990 exploitant la DRV8838.

La bibliothèque Romi32U4 offre des fonctions qui permet de contrôler facilement les moteurs (voir "Programmer avec avr-gcc et AVRDUDE").

Les connexions de la carte contrôleur Romi 32U4 expose des connecteurs prévu pour interfacer une paire d'encodeur rotatif pour Romi https://shop.mchobby.be/product.php?id_product=1452 lien pololu https://www.pololu.com/product/3542 , connecteurs présent par paire (un rangée vers "l'intérieur" et une rangée vers "l'extérieur"). Une paire de connecteurs femelles bas-profile est inclus avec la carte contrôleur Romi 32U4, connecteurs qui peuvent être soudés sur chaque côté de la carte Romi (soit dans la rangée de connecteur "intérieur", soit sur le connecteur "extérieur"). **Note:** les connecteurs doivent être soudés dans la position correspondant au connecteur mâle présent sur l'encodeur rotatif).

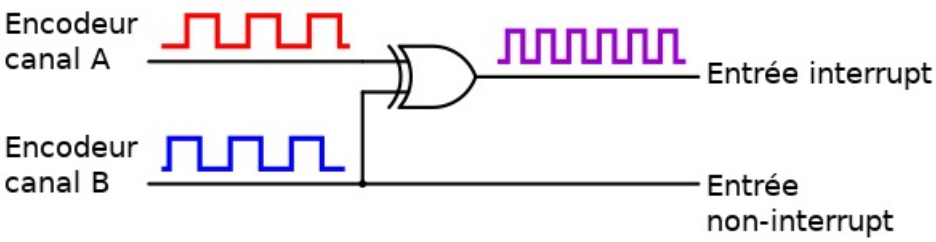
Au fur et à mesure que les piles se déchargent, la tension VSW fournie au contrôleur moteur diminuera, ce qui ralentira le moteur. Il est tout à fait possible de gérer cela dans le code en surveillant la tension des piles (voir section alimentation) **OU** en utilisant des encodeurs (ou autre capteurs) capable de surveiller les mouvements du robot.

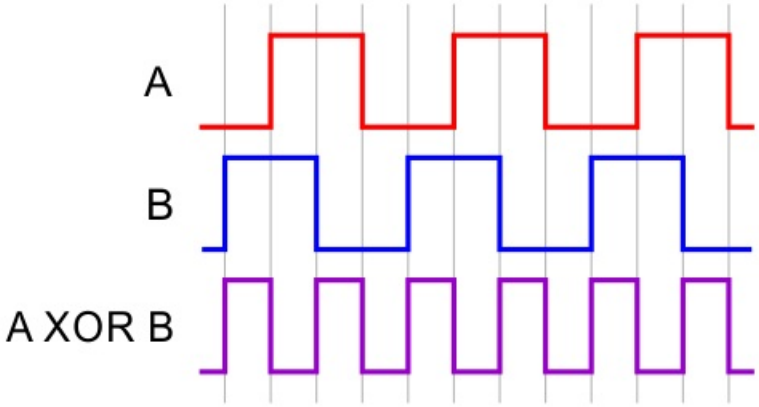
Encodeurs à quadrature

La carte Contrôleur Romi 32U4 est configurée pour connecter les sorties des encodeurs à quadrature du kit "Encodeur pour Romi" sur le microcontrôleur ATmega32U4. Les encodeurs peuvent être utilisé pour surveiller la vitesse de rotation et le sens de rotation des roues du robot. Ils fournissent une résolution de comptage de 12 événements par révolution de l'axe moteur (il faut compter les flan montant et descendant sur les deux canaux), ce qui correspond approximativement à un comptage de 1440 événements par révolution d'une roue Romi (roue montée derrière une boîte de réduction). Voyez les détails de la page produit Paire d'encodeur rotatif pour Romi - CPR 12 https://shop.mchobby.be/product.php?id_product=1452 pour plus de détails.

Les transitions d'un encodeur à quadrature sont souvent détectés en surveillant directement les deux canaux de l'encodeur. Cependant, sur un Romi, ces transitions sont produites à un fréquence très élevée (plusieurs milliers par secondes) lorsque les moteurs fonctionnent, il est nécessaire d'utiliser les interruptions sur les broches de l'AVR ou interruption externe pour lire les encodeurs. Pour réduire le nombre de broches d'interruption nécessaires, la carte de contrôle Romi 32U4 fait une opération XOR des deux canaux de chaque encodeur et connecte le signal résultat sur une broche d'interruption, alors que le canal B de chaque encodeur est connecté sur une broche d'entrée non-interrupt:

- **Broche digital 7** (ou PE6) lit le **signal XOR de l'encodeur droit** en utilisant l'interruption externe INT6 (*external interrupt*).
- **Broche digital 8** (ou PB4) lit le **signal XOR de l'encodeur gauche** en utilisant l'interruption PCINT4 (*change interrupt*).
- **Digital pin 23** (broche analogique 5 ou PF0) pour lire l'encodeur **le canal B de l'encodeur droit**.
- La broche **PE2** lit **la canal B de l'encodeur gauche**.





Le signal en XOR et le signal du canal B peut être utiliser pour reconstruire le signal du canal A avec une nouvelle operation XOR: **(A XOR B) XOR B = A**. Pour les deux encodeurs, la canal B précède le canal A lorsque le moteur tourne en marche avant; Cela signifie que, le signal B monte avant que A monte ET que le signal B descendant avant que A descende. Le diagramme ci-dessous est produit lors d'une rotation vers l'avant.

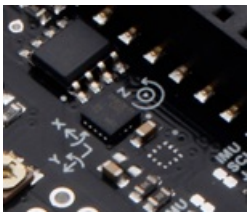
> La description designe les signaux A et B comme libellé sur la carte de contrôle, qui place le signa A à l'avant des deux côtés de la carte.

La bibliothèque Romi32U4 fournir les routines et fonctions popur lire les encodeur et maintenir les indexes de comptage (voir section "Bibliothèque Arduino Romi 32U4").

Capteurs Inertiels

Capteur Intertiel

La carte contrôleur du Romi 32U4 inclus des capteurs inertiels connectés directement sur le bus I2C de l'ATmega32U4. Ces capteurs peuvent être utilisé pour des applications avancées comme la détection de collision ou évaluer l'orientation du robot. Ces capteurs issus du LSM6DS33 de ST lien pololu https://www.pololu.com/product/2736 combinent un accéléromètre 3-axes et un gyroscope 3-axes dans un seul et même composant.



Par défaut, ce capteur inertiel est connecté sur sur le bus I2C de la carte mais il est possible de le déconnecter en sectionnant les prises des cavaliers CMS portant la mention de "IMU SDA Jmp" et "IMU SCL Jmp". Des convertisseurs de niveau lofique (*Level shifters*) présents sur la carte permet à l' ATmega32U4, fonctionnant sous 5 V, de communiquer avec les capteurs 3.3 V . Si un Raspberry-Pi est connecté sur la carte de contrôle, ses broches I²C sont connectées sur le côté 3.3 V du bus I2C (donc derrière le convertisseur de niveau logique). Voir la section sur l'interface Raspberry Pi pour plus d'informations

Popolu recommande une lecture attentive de la fiche technique du LSM6DS33 https://www.pololu.com/file/0J1087/LSM6DS33.pdf (1MB, pdf) pour comprendre comment ces capteurs fonctionnent et comment les utiliser.

Utiliser la capteur

La bibliothèque Romi32U4 (voir la section "Bibliothèque Arduino Romi 32U4") inclus des programmes d'exemples pouvant être utilisés pour découvrir l'usage des capteurs.

Alimentation

Sommaire

Introduction

La carte de contrôle Romi 32U4 inclus des bornes de connexion pour accéder au l'alimentation offerte par le compartiment à pile (6 piles AA). Pololu recommande d'utiliser des piles AA rechargeables NiMH, qui offrent une tension nominale de 7.2 V (1.2 V per pile). Vous pouvez également utiliser des piles Alcaline qui offre, dans ce cas, une tension nominale de 9 V.

Le pôle négatif des piles est connecté sur la masse/GND. Le pole positif des piles est désigné par la mention **VBAT**. VBAT passe par une diode de protection (contre la polarisation inverse) et alimente ensuite le convertisseur de tension contrôlé par le bouton (ou interrupteur) présente sur la carte. La sortie du circuit d'alimentation est identifié à l'aide de **VSW** (*Volt SWitch* pour alimentation avec interrupteur).

VSW fournit la puissance aux moteurs par l'intermédiaire du contrôleur moteur DRV8838, de sorte que les moteurs peuvent uniquement fonctionner que si les piles sont installées et que l'interrupteur d'alimentation est fermé.

La diode de protection et tension disponible sur VSW (après l'interrupteur) peut être surveillé à l'aide d'un pont diviseur de tension connecté sur la **broche analogique 1** (PF6) par défaut. Le diviseur de tension sort une tension qui est égale au 1/3 de tension des piles, ce qui sera en dessous de la tension maximale supportée par une broche analogique de l'ATmega32U4 (soit 5 V) pour autant que la tension des piles reste sous 15 V (a noter que la tension maximale du contrôleur moteur DRV8838 est limité à 10.8V). La fonction `readBatteryMillivolts()` de la bibliothèque Romi32U4 peut être utilisée pour déterminer la tension des piles à partir de la lecture. Le cavalier libellé “A1 = BATLEV” peut être coupé pour pour déconnecter le diviseur de tension et ainsi libérer une broche sur le microcontrôleur.

Interrupteur d'alimentation (le circuit)

La carte de contrôle Romi 32U4 utilise circuit de mise en marche/arrêt de puissance à bouton poussoir de Pololu lien pololu https://www.pololu.com/product/2808 , qui offre un interrupteur de puissance à état solide pour votre robot avec un bouton poussoir présent sur la carte. Par défaut, ce bouton poussoir peut être utilisé pour contrôler l'alimentation: pousser une fois pour activer l'alimentation, pousser une autre fois pour couper l'alimentation. En alternative, un autre bouton poussoir peut être connecté sur les broches **PWRA** et **PWRB** pour être utilisé comme bouton marche/arrêt. Plusieurs boutons peuvent être branchés en parallèle pour avoir plusieurs points de commande et chacun des boutons sera capable de commander le mise en marche/arrêt de la carte. **Le circuit de *latching* effectue un déparasitage du bouton mais un rebond excessif (plusieurs ms) le fonctionnera pas correctement avec ce circuit**.

Pour un fonctionnement correcte du bouton poussoir, l'interrupteur de la carte devrait être laissé sur la position **Off**. (Faire glisser l'interrupteur sur la position "On" pour basculer la carte en position marche et l'interrupteur doit revenir en position "Off" avant de pouvoir désactiver la carte à l'aide du bouton poussoir.)

En alternative, vous pouvez désactiver le bouton poussoir en courant le cavalier libellé **Btn Jmp**; cela transfert le contrôle de la puissance à l'interrupteur présent sur la carte. Un interrupteur séparé peut être connecté sur la broche **GATE** et être utilisé à la place.

Le circuit de contrôle marche/arrêt offre plusieurs connexions alternative permettant de réaliser un push-on-only (*bouton pour marche uniquement*) ou push-off-only (*bouton pour arrêt uniquement*). D'autres broches d'entrées permettent d'activer des options complémentaires comme permettre à votre robot de couper sa propre alimentation. Ces iotuibs de contrôle avancé sont disponibles par l'intermédiaire des connexions du bouton et 4 entrées de contrôle:

Broche	Description
PWRA	Connecté via un bouton momentané sur la broche "PWRB" pour la fonctionnalité standard push-on/push-off (*pousser-marche/pousser-arrêt*). Connecté via un bouton momentané à la masse pour la fonctionnalité on-only (*bouton pour marche uniquement*).
PWRB	Connecté via un bouton momentané sur la broche "PWRA" pour la fonctionnalité standard push-on/push-off.
ON	Une impulsion sur cette broche (> 1 V) passe le circuit de puissance en état marche. Cette broche est utilisable que lors du fonctionnement de la carte en mode bouton poussoir (*le cavalier du bouton n'a pas été coupé*).
OFF	Une impulsion sur cette broche (> 1 V) passe le circuit de puissance en état arrêt (ex: permet à un circuit de couper sa propre alimentation). *Cette broche ne fonctionne que si la fonctionnalité bouton poussoir pour marche/arrêt est activé*.
CTRL	Avec la fonctionnalité bouton poussoir pour marche/arrêt est activé, cette broche détermine directement l'état du circuit d'alimentation (marche ou arrêt). Une impulsion niveau haut (> 1 V) sur cette broche active le circuit de puissance; une impulsion au niveau bas (ex: forcer la broche au niveau bas avec un sortie microcontrôleur ou pousser le bouton poussoir connectant cette broche à la masse) aura pour effet de désactiver le circuit de puissance. Laisser cette broche déconnectée ou flottant lorsqu'elle ne doit pas être pilotée. Note: cette broche ne peut pas être placée au niveau haut en même temps que la broche "OFF".
GATE	Avec la fonctionnalité du bouton poussoir désactivé (cavalier du bouton coupé), cette broche contrôle l'état du circuit de puissance: placer au niveau bas cela activera le circuit de puissance, tandis que laissé flottante cela désactivera le circuit de puissance. Connecter à la masse par l'intermédiaire d'un interrupteur pour un fonctionnement en marche/arrêt. Laisser cette broche flottante ou déconnecter pour un fonctionnement marche/arrêt avec le bouton poussoir. Pololu recommande de piloter cette broche au niveau bas uniquement ou la laisser flottante; cette broche ne devrait jamais être placée au niveau haut lorsque le l'interrupteur est en position "On" (marche).

Régulateurs 5V et 3.3V

La connexion VSW fourni l'alimentation au régulateur 5 V, dont la sortie est désignée par **VREG**. La tension des piles est régulée vers 5 V par un régulateur MP4423H (hacheur de type buck). Même si le régulateur accepte une tension jusqu'à 36 V, le contrôleur moteur accepte une tension maximale de 10.8 V. Lorsque disponible, VREG est généralement utilisé pour alimenter l'ATmega32U4, contrôleur moteur et les encodeurs. Le restant du courant disponible sur le régulateur, qui dépends de la tension d'entrée et conditions ambiantes, peut être utilisé pour alimenter d'autres périphériques; cela peut inclure un Raspberry Pi branché sur la carte (consommant généraleent quelques centaines de milliampères).

Dans les conditions normales, un courant allant jusque 2 A peut être disponible sur la sortie VREG. (Ce régulateur lien pololu https://www.pololu.com/product/2858 Pololu est également disponible sous la forme d'un circuit intégré.)

Le régulateur MP4423H porpose une sortie `Power Good` (puissance OK) sous forme d'une sortie à drain ouvert, la broche **PG** nécessite une résistance pull-up externe. PG est placé à la masse lorsque la sortie du régulateur 5V tombe en dessous de 85% de la tension nominale et offre une grande impédance lorsque la tension de sortie dépasse les 90% de la tension nominale. Le circuit de régulation de la carte de contrôle Romi 32U4 peut être désactivé en placant la broche *shutdown* du régulateur, **REGSHDN**, au niveau haut; Cela permettra à l'étage logique de la carte de contrôle d'être alimenté par la connexion USB (si disponible).

La carte de contrôle Romi 32U4 dispose également d'un régulateur 3.3 V LDO (*Low Drop Out* = faible chute de tension) qui est alimenté depuis la sortie du circuit de sélection d'alimentation de l'étage logique (décrit ci-dessous). La sortie du régulateur 3.3 V est désigné sous la mention **3V3** qui est utilisée pour alimenter les capteurs inertiels de la carte et les *level shifters* (circuit de conversion de niveau logique).

Sélection de l'alimentation de la logiqueLogic power selection

Le circuit de sélection d'alimentation de la carte Romi 32U4 utilise le multiplexeur d'alimentation TPS2113A lien pololu https://www.pololu.com/product/2596 de Texas Instruments pour choisir sa source d'alimentation 5V (désignée **5V**). Cette alimentation 5V peut être fournie depuis l'USB ou le bloc pile (via régulateur 5V), le TPS2113A peut passer d'une source d'alimentation à l'autre en toute sécurité et sans interruption. Le TPS2113A est configuré pour sélectionner en priorité sa source depuis le bloc pile (VREG) à moins que la sortie du régulateur tombe sous 4.5 V. Si cela arrive, le multiplexeur d'alimentation sélectionne la plus haute des deux sources, qui est généralement la tension du bus USB si la carte de contrôle est connecté en USB.

Par conséquent, lorsque la carte de contrôle Romi 32U4 est connectée sur un ordinateur via USB, il recevra une alimentation 5 V sur sa logique de contrôle **même si l'interrupteur d'alimentation est en position arrêt. Cela est très utile pour téléverser et tester un programme sans consommer les piles et sans faire fonctionner les moteurs. Il est tout à fait possible d'avoir la carte USB connectée en même temps que le bloc pile actif (interrupteur en position marche sans que cela ne pose de problèmes.**

La source d'alimentation est indiquée par l'état de la broche **STAT**; cette broche est une sortie à drain ouvert qui est au niveau bas si la source d'alimentation est externe et à haute impédance si l'alimentation USB est sélectionnée. La limite de courant du TPS2113A est fixée à un courant normila de 1.9 A. Plus d'information sur le multiplexeur d'alimentation peut être obtenu depuis la fiche technique du TPS2113A https://www.pololu.com/file/0J771/tps2113a.pdf (1 Mio pdf).

La sortie 5 V du circuit de sélection d'alimentation est utilisé pour alimenter le microcontrôleur ATmega32U4 de la carte de contrôle, la logique du contrôleur moteur DRV8838 motor drivers et les encodeurs; il peut également alimenter un Raspberry-Pi connecté sur la carte.

Alimentation Raspberry Pi

Par défaut, la carte contrôleur fournira l'alimentation au Raspberry-Pi depuis sa ligne d'alimentation 5V. Dans cette situation, Pololu recommande de mettre le circuit d'alimentation en marche de sorte que le Raspberry-Pi soit alimenté depuis les piles par l'intermédiaire du la carte contrôleur (et son régulateur DC/DC). En alternative, il est aussi possible d'utiliser un bloc d'alimentation USB pour alimenter l'ensemble via le connecteur USB présent sur la carte contrôleur (a noter que l'AVR peut expérimenter un reset intempestif lorsque lorsque le Raspberry-Pi est alimenté dans cette configuration). Les ports USB des ordinateurs ne sont généralement pas dimensionnés pour fournir le courant nécessaire au fonctionnement d'un contrôleur Romi 23U4 + Raspberry Pi branché dessus.

L'alimentation fournie au Raspberry-Pi peut être désactivée en plaçant la broche **RPISHDN** (Raspberry Pi shutdown) à 5V.

Le circuit de diode (idéal) de la carte contrôleur permet de connecter, en toute sécurité, différentes sources d'alimentation sur le Raspberry Pi (par exemple, via le connecteur microB du Raspberry Pi) tandis que la carte de contrôle est connectée et alimentée. Plus clairement, il est possible d'avoir n'importe quelle combinaison d'alimentation via la carte de contrôle (via USB), alimentation pile, le Raspberry-Pi (via microUSB). La broche **RPI5V** offre un accès direct au rail 5V du Raspberry-Pi, qui sera généralement la tension la plus haute des deux sources d'alimentationi. La tension de sortie 3.3V du Raspberry Pi est également disponible sur un la broche **RPI3V3** .

Le circuit de diode empêche l'alimentation de passer d'un circuit d'alimentation à l'autre (dans le mauvais sens): Le Raspberry Pi ne peut pas fournir une alimentation logique 5 V à la carte de contrôleur par l'intermédiaire du connecteur 40 broches.

Distribution d'alimentation

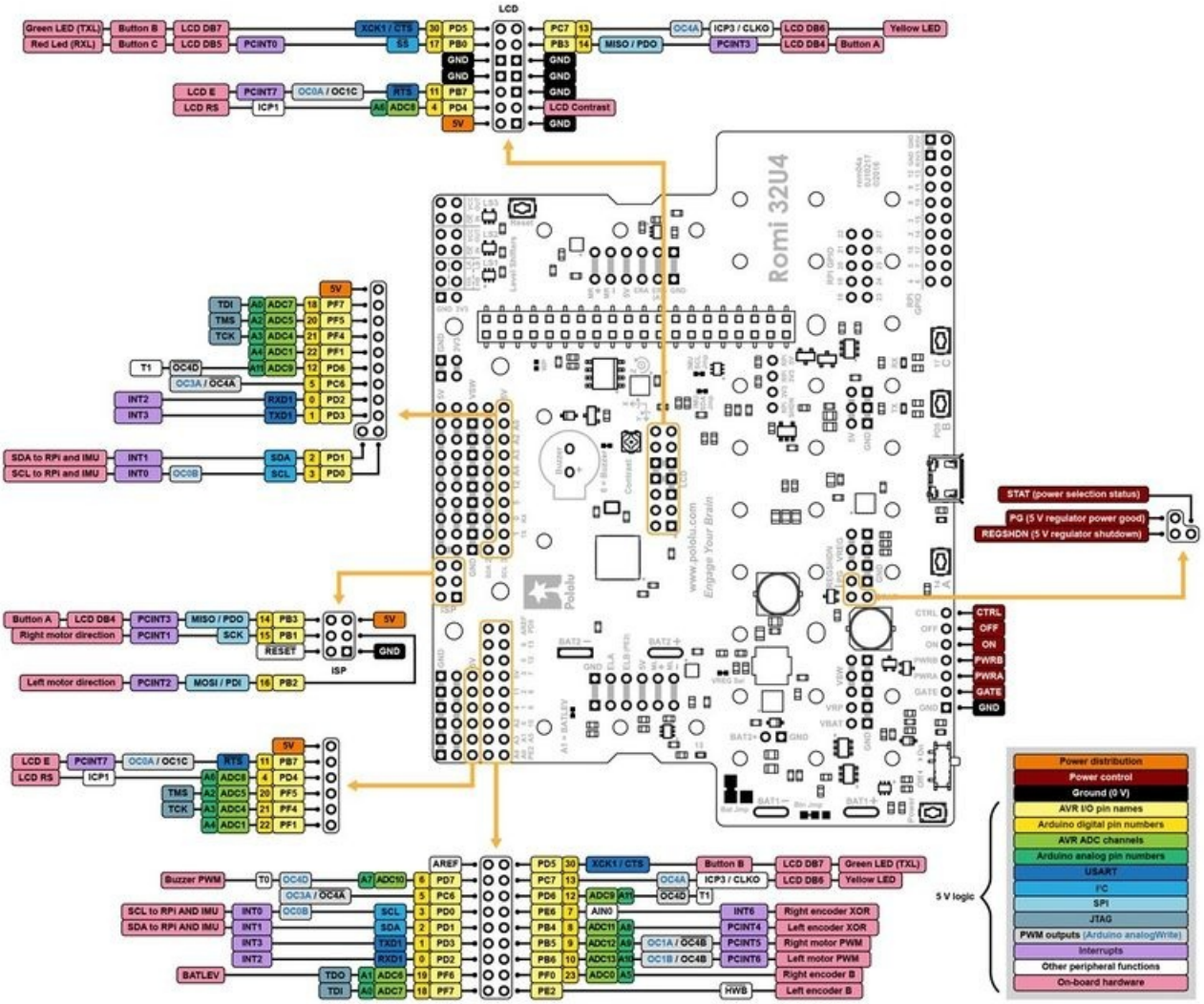
- **VBAT** est connecté sur le contact de pile libellé **BAT1+** et offre une connexion directe sur l'alimentation pile.
- **VRP** offre un accès à la tension des piles après la diode de protection (contre polarisation inverse).
- **VSW** offre un accès à la tension des piles après la diode de protection et après circuit de coupure.
- **VREG** est la tension 5V en sortie du régulateur (VRegulator).
- **5V** est la sortie 5V après le circuit multiplexeur d'alimentation TPS2113A qui, par défaut, est connecté sur VREG, mais bascule sur l'alimentation 5V USB si VREG chute trop bas.
- **3V3** est la sortie du regulateur 3.3 V LDO (*Low Drop Out*, à faible chute de tension).

Voir la section sur "les extensions" pour un diagramme des bus de distribution d'alimentation (et points d'accès).

Extension

La carte contrôleur Romi 32U4 propose plusieurs surface d'extension (d'abord trois groupes près de la face avant, du milieu et face arrière) qui font des breakouts de tous les GPIO (*General Purpose Input Output* lines) du microcontrôleur ATmega32U4 et du Raspberry Pi. La carte offre également un accès aux différentes sources d'alimentation, sorties, broches de contrôle et bus disponibles pour faciliter les connexions. Les diagrammes suivants permet d'identifier la location de ces broches et la matériel qui y est branché/rattaché; ces diagrammes sont également disponibles sur un PDF en version imprimable https://www.pololu.com/file/0J1261/romi-32u4-control-board-pinout-power.pdf (1Mio pdf). Pour plus d'informations sur le microcontrôleur ATmega32U4 et ses périphériques, n'hésitez pas à consulter la documentation officielle de l'Atmel ATmega32U4.

Brochage de la carte contrôleur Romi 32U4 (broches ATmega32U4, périphériques et contrôle de puissance)



Brochage de la carte contrôleur Romi 32U4 (broches Raspberry Pi, périphériques et *level shifters*).

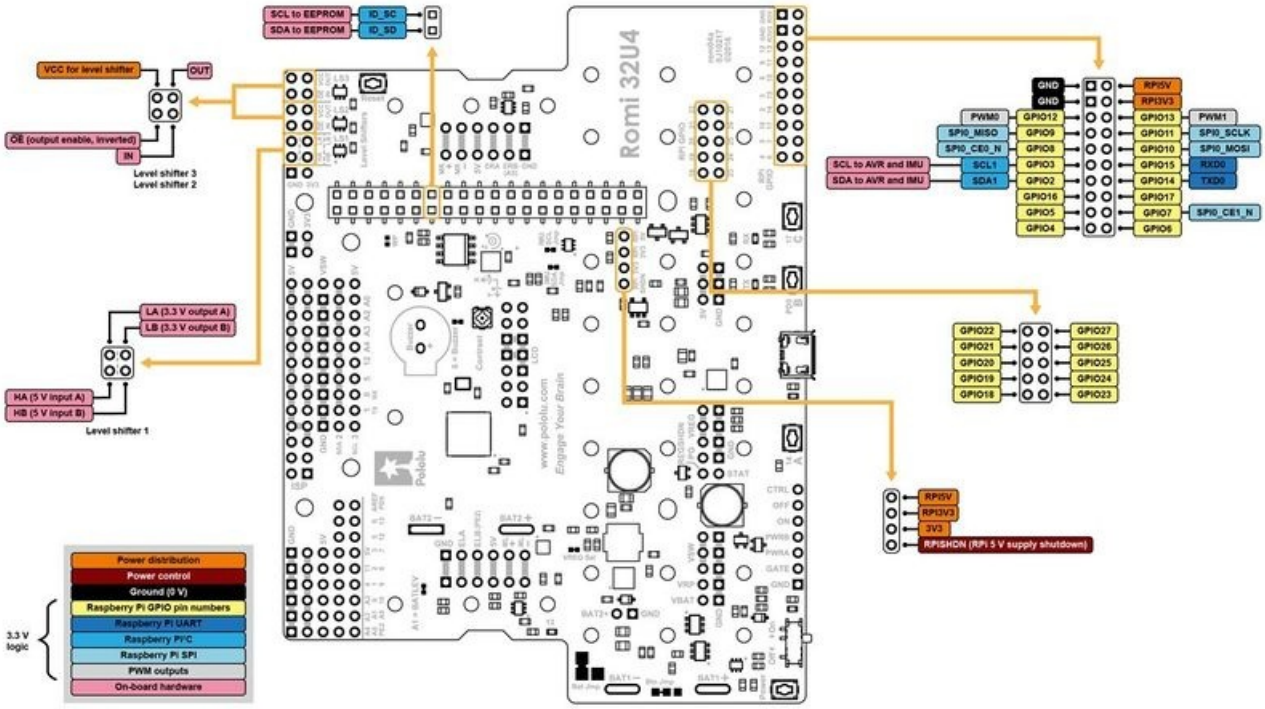
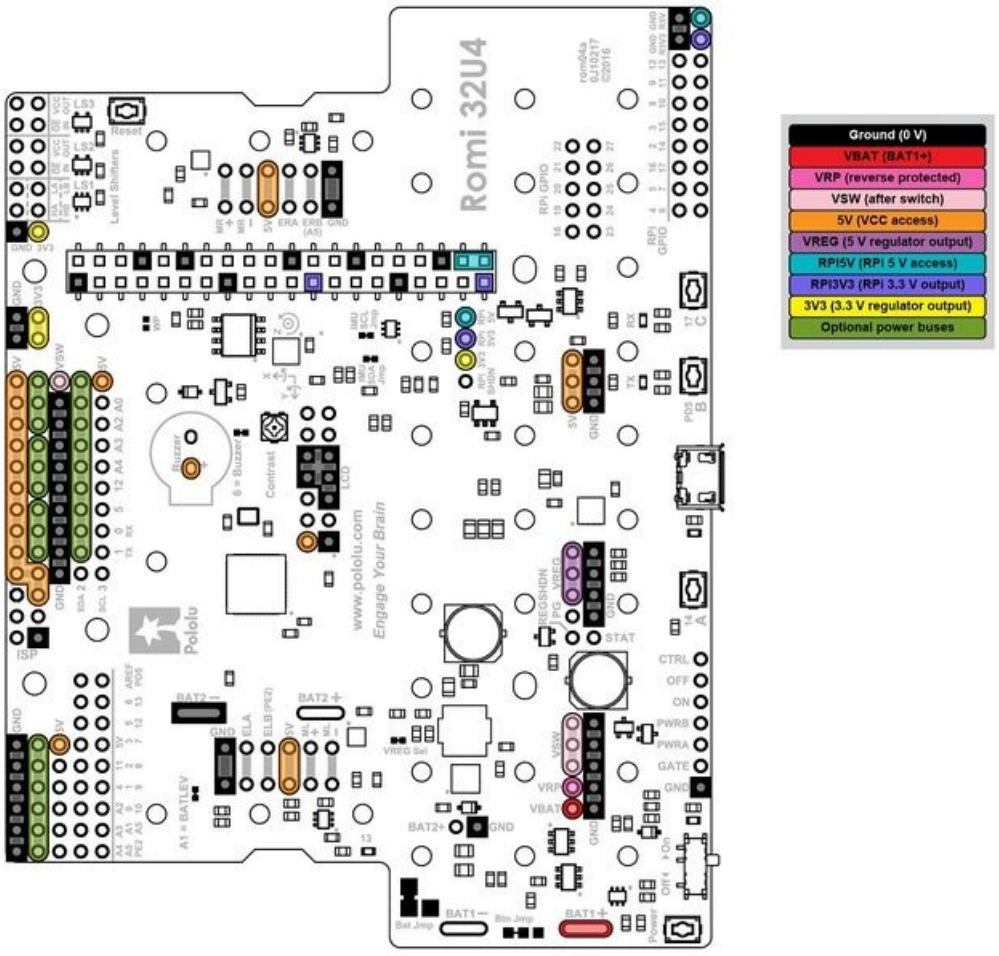


Diagramme de distribution d'alimentation de la carte contrôleur Romi 32U4.



Interface Raspberry Pi

Sommaire

Interface Raspberry Pi

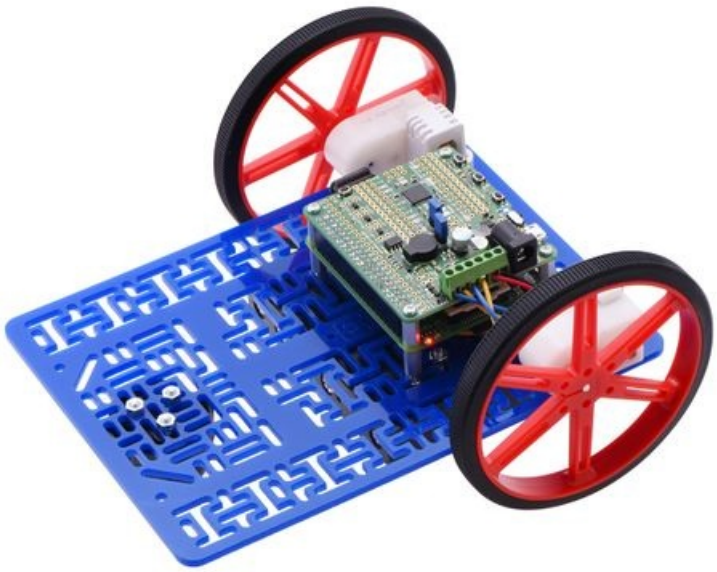
Pour faciliter l'interfaçage avec les nano-ordinateurs Raspberry Pi afin de pouvoir étendre les capacités de traitement du Romi. Il dispose d'un connecteur et de trous de montage correspondant au spécifications Raspberry Pi HAT (*Hardware Attached on Top*). Le Romi est conçu pour être connecté sur un GPIO 40 broches d'un modèle B+ et plus récent (incluant Raspberry Pi 3/4 Modèle B et Modèle A+). Un connecteur femelle 2×20 broches (empattement 2.54mm) est soudé sur la carte de contrôle, la carte de contrôle est par ailleurs livrée avec 4 entretoise, visserie et écrou.

Communication I2C

Lorsqu'elle est utilisée avec un Raspberry Pi, la carte de contrôle est conçue pour agir comme un périphérique auxiliaire communiquant avec le Raspberry Pi via l'interface I2C (également connue sous le nom *2-wire Serial Interface*, TWI ou interface série 2 fils). Dans pareil cas, les lignes de données (SDA) et d'horloge (SCL) du bus I2C du microcontrôleur ATmega32U4 sont connectées sur leurs lignes correspondantes du Raspberry Pi par l'intermédiaire de *level-shifter*. Ces convertisseurs de niveaux logiques (dit *level shifters*) convertissent la tension logique 5V de l'AVR vers la tension logique 3.3V du Raspberry Pi (et vice-versa).

Pololu à écrit un bibliothèque Arduino https://github.com/pololu/pololu-rpi-slave-arduino-library pour la famille de carte 32U4 pour permettre au microcontrôleur d'agir comme un périphérique esclave I2C (dit *suiveur* suivant la nouvelle terminologie I2C) et le Raspberry Pi comme périphérique maître I2C (aussi dit *leader*).

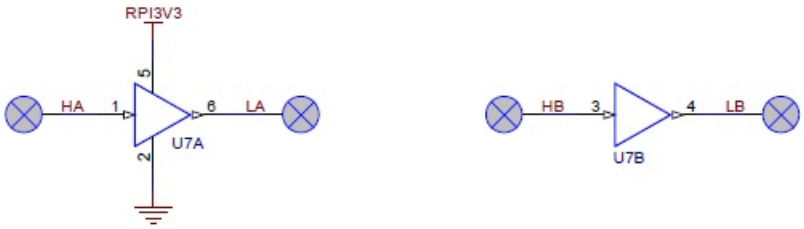
Un tutorial du blog Pololu https://www.pololu.com/blog/577 (*Anglais*) démontre l'usage de cette bibliothèque avec des codes d'exemples permettant ainsi de contrôler/surveiller le robot à distance à l'aide d'un serveur WEB installé sur le Raspberry Pi. Ce tutoriel utilise le lien pololu https://www.pololu.com/product/3119 et un châssis coupé au laser. Ces instruction de configuration du Raspberry Pi et bibliothèque esclave pour Arduino sont applicables à la carte de contrôle 32U4 pour Romi (Pololu devrait publier un tutoriel adhoc dans le futur, si ce n'est déjà fait).



level shifters à usage général

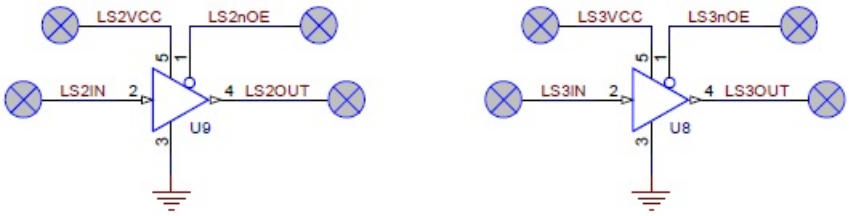
En plus des *Level Shifters* dédiés au bus I2C, la carte propose quelques *level shifters* libre d'usage puisqu'ils ne sont pas encore connectés sur des signaux.

LS1 est un *level shifter* deux canaux uni-directionnels qui converti une paire d'entrée 5V (**HA** et **HB**) vers la paire de sortie 3.3 V (**LA** et **LB**).



LS2 et **LS3** sont des canaux *level shifter* uni-directionnel, tristatable (3 états). Chacun d'entre eux expose 4 broches: $\overline{OE}$ (output enable), **IN** (input), **OUT** (shifted output) et **VCC** (alimentation de la logique).

- Lorsque $\overline{OE}$ est au niveau BAS, OUT présente une sortie à haute impédance.
- Lorsque $\overline{OE}$ est au niveau BAS, OUT propose l'état correspondant à **IN** (avec la tension appliquée sur VCC).



Par exemple: si vous placez $\overline{OE}$ au niveau BAS, connectez un signal 3.3V (HAUT) sur IN, et 5V sur VCC alors le signal IN sera converti en logique 5V sur la sortie OUT.

Le niveau logique d'entrée IN peut aller de 1.8V à 5.5 V. L'alimentation VCC (et niveau logique de sortie) peut aller de 3V à 5.5V. Le signal d'entrée IN peut avoir un niveau logique inférieur ou supérieur à la tension VCC: il est donc possible de connecter un signal 5V sur une entrée (IN), 3.3V sur VCC. Il est également possible d'avoir un signal 3.3V en entrée et VCC placé à 5V.

Alimenter le Raspberry Pi depuis la carte contrôleur

Par défaut, la carte contrôleur Romi fournira une tension d'alimentation 5V au Raspberry-Pi branché sur le connecteur GPIO.

Voir la section alimentation pour plus de détail sur le partage d'alimentation, et leur contrôle, entre les deux cartes.

ID EEPROM

La carte contrôleur Romi 32U4 inclus une EEPROM de 32 Kilobits (4096 octets) qui est sir les lignes Raspberry Pi ID_SD et ID_SC pins. Le contenu de l'EEPROM n'est pas initialisé mais il est possible d'y programmer un ID EEPROM en suivant le format des spécifications Raspberry Pi HAT https://github.com/raspberrypi/hats (*RaspberryPi.org, anglais*) et les utilitaires mentionnés dans ces spécifications. Lorsqu'elle est correctement programmée, l'EEPROM permet au Raspberry Pi d'identifier la carte connectée et de s'auto configurer pour l'utiliser (dans le cas présent, il s'agit de la carte de contrôle Romi 32U4).

La protection d'écriture de l'EEPROM peut être activée en utilisant une pointe de soudure pour ponter le cavalier labellisé "WP" (à côté de l'EEPROM). Par défaut, cette EEPROM n'est pas en lecture seule.

Broches

Assignation des broches

La table ci-dessous liste l'assignation des broches ATmega32U4 exploité sur la carte contrôleur Romi 32U4.

Cette information est utile si vous désirez ajouter vitre propre électronique sur le Romi 32U4, écrire du code de bas niveau pour le matériel interfacé ou si vous voulez simplement mieux comprendre comment fonctionne le Romi 32U4. Chaque ligne représente une broche physique du microcontrôleur ATmega32U4.

la colonne "ATmega32U4 broche" montre le nom officiel sur l'ATmega32U4 tel que repris sur fiche technique de l' ATmega32U4 https://www.microchip.com/wwwproducts/en/ATmega32u4 .

La colonne "Arduino broche" reprend le nom tel que connu dans l'environnement Arduino. Ces noms peuvent généralement être utilisé comme argument pour les fonction Arduino nécessitant un numéro de broche. Il y a cependant quelques exceptions. Par exemple, passer ne numéro 4 à `analogRead` lira la broche A4 et nom la broche 4. Il y a également des limitations matérielles autorisant des fonctions à ne travailler qu'avec un ensemble limité de broches.

La colonne "Romi 32U4 fonctions" indique l'assignation de la broche sur la carte de contrôle Romi 32U4. De nombreuses broches peuvent servir à plusieurs utilisation (en changeant de mode). Par exemple, PB0 peu lire l'état du bouton C lorsqu'elle est en entrée et en sortie, peut contrôler la LED rouge ou servir de ligne de donnée (LCD data) pour un afficheur LCD.

La colonne "Note/fonction alternative" documente d'autres fonctionnalités de la broche, même si certaines fonctionnalités seraient raisonnablement impossible à exploiter.

ATmega32U4 broche	Arduino broche	Romi 32U4 fonctions	Notes/fonctions alternatives
PB7	11	Ligne contrôle LCD (E)	Timer0 PWM output A (OC0A) Timer1 PWM output C (OC1C) Contrôle flux UART (RTS) Pin-change interrupt (PCINT7)
PD4	4, A6, 24	Ligne contrôle LCD (RS)	Entrée analogique (ADC8) Timer1 input capture pin (ICP1)
PB3	14, MISO	bouton utilisateur A Ligne donnée LCD DB4	SPI MISO Pin-change interrupt (PCINT3) Ligne programmation ISP (PDO)
PB0	17, LED_BUILTIN_RX, SS	LED Rouge (RX) Bouton utilisateur C Ligne donnée LCD DB5	SPI selection esclave (SS) Pin-change interrupt (PCINT0)
PC7	13, LED_BUILTIN	LED Jaune Ligne donnée LCD DB6	Timer4 PWM output A (OC4A) Timer3 input capture pin (ICP3) Divided system clock output (CLKO)
PD5	30, LED_BUILTIN_TX	LED verte (TX) Bouton utilisateur B Ligne donnée LCD DB7	UART horloge externe (XCK1) UART contrôle flux (CTS)
PD7	6, A7, 25	PWM Buzzer	Entrée analogique (ADC10) Timer4 PWM output D (OC4D)

			Timer0 counter source (T0)
PF6	A1, 19	Entrée tension piles (VIN/3)	Entrée analogique (ADC6) JTAG test data out (TDO)
PB6	10, A10, 28	PWM moteur gauche	Entrée analogique (ADC13) Timer1 PWM output B (OC1B) Timer4 PWM output B (OC4B) Pin-change interrupt (PCINT6)
PB2	16, MOSI	Direction Moteur gauche	SPI MOSI Pin-change interrupt (PCINT2) ISP ligne programmation (PDI)
PB5	9, A9, 27	PWM moteur droit	Entrée analogique (ADC12) Timer1 PWM output A (OC1A) Timer4 PWM output B (OC4B) Pin-change interrupt (PCINT5)
PB1	15, SCK	Direction moteur droit	SPI Horloge (SCK) Pin-change interrupt (PCINT1) Ligne programmation ISP (SCK)
PB4	8, A8, 26	Entrée XOR de l'encodeur à gauche	Entrée analogique (ADC11) Pin-change interrupt (PCINT4)
PE2	-	Entrée encodeur à gauche	Hardware bootloader select (HWB)
PE6	7	Entrée XOR de l'encodeur à droite	Entrée négative du comparateur analogique (AIN0) External interrupt source (INT6)
PF0	A5, 23	Entrée encodeur à droite	Entrée analogique (ADC0)
PD0	3, SCL	Horloge I2C pour communication Raspberry Pi et capteurs inertiels	Timer0 PWM output B (OC0B) External interrupt source (INT0)
PD1	2, SDA	Donnée I2C pour communication Raspberry Pi et capteurs inertiels	External interrupt source (INT1)
PD2	0	**entrée/sortie disponible**	UART réception (RXD1) External interrupt source (INT2)
PD3	1	**entrée/sortie disponible**	UART émission (TXD1) External interrupt source (INT3)
PC6	5	**entrée/sortie disponible**	Timer3 PWM output A (OC3A) Timer4 PWM output A (OC4A)
PD6	12, A11, 29	**entrée/sortie disponible**	Entrée analogique (ADC9) Timer4 PWM output D (OC4D) Timer1 counter source (T1)

PF7	A0, 18	**entrée/sortie disponible**	Entrée analogique (ADC7) JTAG test data in (TDI)
PF5	A2, 20	**entrée/sortie disponible**	Entrée analogique (ADC5) JTAG test mode select (TMS)
PF4	A3, 21	**entrée/sortie disponible**	Entrée analogique (ADC4) JTAG test clock (TCK)
PF1	A4, 22	**entrée/sortie disponible**	Entrée analogique (ADC1)
RESET	-	Bouton Reset	Résistance Pull-up interne, activé au niveau BAS
AREF	-	-	Référence analogique

Ajouter de l'électronique

Sommaire

Ajouter de l'électronique

Cette section contient des trucs et astuces pour connecter des éléments électroniques complémentaires sur la carte de contrôle Romi 32U4.

Les broches I/O encore libre

Si vous voulez ajouter des périphériques complémentaires (ou recevoir des informations depuis l'AVR), vous aurez besoin de les connecter sur une ou plusieurs broches I/O de l'AVR. La section sur l'assignation des broches liste de toutes les broches et leur utilisation. De nombreuses entrées/sorties sont déjà utilisées par la carte mais **il y a encore 8 broches I/O disponibles: 0, 1, 5, 12, A0, A2, A3 et A4**. Toutes ces broches I/O disponibles peuvent être utilisées en entrée ou en sortie; chaque broche dispose également de fonctionnalités spéciales.

Broche 0 (PD2) et broche 1 (PD3) sont les lignes RX et TX du port série TTL de l'AVR.

Broche 5 (PC6) est une sortie PWM matérielle et utilisable avec la fonction Arduino `analogWrite()`. La broche 12 (A11/PD6) peut aussi être utilisée comme sotie PWM mais n'est pas supportée par `analogWrite()`, et utiliser la broche 12 en PWM pourrait entrer en conflit avec la broche 6 (qui contrôle le Buzzer) puisque ces deux broches sont des sorties complémentaire du Timer4 canal D.

Les broches 12 (A11/PD6), A0 (18/PF7), A2 (20/PF5), A3 (21/PF4) et A4 (22/PF1) peuvent être utilisée comme entrées analogiques.

Libérer plus de broches I/O

Si les entrées/sorties disponibles (libre d'usage) ne sont pas suffisants pour connecter les périphériques dont vous avez besoins, il est possible de désactiver (ou déconnecter) quelques fonctionnalités de la carte contrôleur Romi 32U4 pour libérer des I/O supplémentaires.

Si vous n'avez pas besoin que l'AVR mesure la tension des piles, vous pouvez utiliser la broches A1 (19/PF6) à d'autres fins. Cette broche peut être utilisée comme entrée/sortie numérique ou comme entrée analogique. Pour utiliser cette broche, il serai nécessaire de couper la piste entre les deux pastilles du cavalier "A1 = BATLEV" de façon à déconnecter la broche du diviseur de tension (du bloc pile). Si vous voulez utiliser A1 en sortie uniquement alors vous pourriez omettre de couper ce cavalier.

Si vous le connectez pas un LCD sur le connecteur LCD alors la broche 11 (PB7) et broche 4 (A6/PD4) son disponibles. ces deux broches peuvent être utilisées comme entrées/sorties numériques. De surcroît, la broche 11 peut être utilisée en sortie PWM (et une interruption "*pin change*") tandis que la broche 4 peut être utilisée comme entrée analogique.

Si vous ne connectez pas d'afficheur LCD sur le connecteur LCD alors vous pouvez utiliser le potentiomètre de contraste pour autre chose. La sortie du potentiomètre offre une tension entre 0 et 5V accessible sur le connecteur de l'afficheur LCD. Il peut être connecté sur n'importe quelle entrée analogique libre pour lire la valeur du potentiomètre sur l'AVR OU vous pouvez connecter la sortie du potentiomètre sur vos propres périphériques additionnels.

Si vous n'avez pas besoin du Buzzer alors vous pouvez libérer la broche 6 (A7/PD7) en coupant la piste entre les deux pastilles du cavalier libellé "6 = Buzzer". La broche 6 peut être utilisée en sortie PWM, ligne d'entrée/sortie numérique. Désactiver le buzzer permet également de liberer le Timer4, qui dispose de plusieurs broches de sortie PWM. Ces broches peuvent être utilisées comme sortie PWM si elles ne sont pas requise pour d'autres tâches normales.

Si vous n'avez pas besoin des encodeurs rotatifs alors vous pouvez libérer les broches 7 (PE6), broches 8 (A8/PB4), PE2 et broche A5 (23/PF0). Chacune de ces broches dispose d'un cavalier CMS dont il est possible de sectionner la piste (entre les deux pastilles) pour déconnecter la sortie Encodeur de l'AVR.

Faites attention lorsque vous connectez des composants sur les broches 13 (PC7), broche 17 (PB0) et broche 30 (PD5). Celle-cis sont utilisées pour contrôler les LEDs du Romi 32U4. Toutes ces broches sont contrôlées en sorties par le bootloader. Les broches 17 (PB0) et 30 (PD5) sont utilisées comme indicateurs RX et TX; par conséquent, si vous envoyez ou recevez des données via USB alors le programme Arduino prend le contrôle de ces broches dans ses routines ISR (interrupt service routines) alors que votre programme est en cours de fonctionnement.

Périphériques I2C

Il est possible de raccorder des périphériques I2C additionnel sur le bus I2C du controleur Romi 32U4 pour autant que les adresses de vos périphériques n'entre pas en conflit avec les périphériques déjà présents sur le bus (le LSM6DS33 qui utilise l'adresse 7-bits 1101011, soit 0x6b). Les broches I2C de l'ATmega32U4’s (2 et 3) fonctionne à 5V. Si vous voulez connecter un périphérique 3.3V alors vous pouvez la connecter sur la partie 3.3V du bus (accessible sur la broche GPIO 2 du connecteur Raspberry Pi pour SDA 3.3V et GPIO 3 du connecteur Raspberry Pi pour SCL. Ces signaux en 3.3V restent accessibles même si le Raspberry Pi n'est pas connecté. Si vous avez besoin de travailler avec d'autres niveaux logiques (ex: 1.5V) alors il sera nécessaire d'ajouter d'autres convertisseurs de niveau logique.

Si vous ne désirez utiliser le capteur inertiel présent sur le bus I2C alors vous pouvez couper les pistes entre les pastilles des cavaliers CMS libellés "IMU SDA Jmp" et "IMU SCL Jmp". Cela libère les broches 2 (PD1) et 3 (PD0) pour une utilisation comme entrée/sortie numérique aussi longtemps qu'il n'y a pas de Raspberry-Pi branché sur la plateforme Romi. A noter que les broches I2C de l'AVR resterons connectées sur les *level shifters* de la carte et seront donc ramenées à +5V (à cause des résistances pulled-up I2C).

Alimentation

Les points d'alimentation de la carte de contrôle Romi sont disponibles en différents emplacement de la carte. Si vous alimentez d'autres périphériques depuis VSW alors ils seront alimentés lorsque l'interrupteur d'alimentation est en position "ON" (marche) et ces périphériques recevrons la tension d'alimentation produite par les piles.

Si vous alimentez vos périphériques depuis VREG alors ils recevrons une tension d'alimentation de 5V lorsque les piles sont installées et l'interrupteur d'alimentation sur la position ON (mais ils ne seront pas alimentés depuis l'USB). Si vous les alimentez depuis la broche 5V alors ils seront alimentés en 5V lorsque la logique de la carte contrôleur Romi sera alimentée. Si vous les alimentez depuis 3V3 alors ils seront alimentés lorsque la logique de la carte de contrôleur Romi est alimentée.

Voir la section alimentation pour plus d'informations sur les points d'alimentation (et le courant qu'ils peuvent délivrer).

Il est aussi possible d'ajouter votre propre interrupteur d'aliment pour pour contrôler l'alimentation de la carte contrôleur Romi 32U4 comme décrit dans la section d'alimentation.

Masse

Vous devriez aussi vous assurer que les masses de votre système sont également connectées. Le point de masse (GND/Ground) du contrôleur Romi 32U4 est accessible sur toutes les broches libellée "GND". Cette masse devrait être connectée sur le point de masse de toutes les cartes (ou capteur) ajouté sur votre robot Romi.

Contrôler un servo

Contrôler des servos

Il est possible de modifier la bibliothèque Servo d'Arduino IDE pour utiliser le Timer 3 à la place du Timer 1 avec une carte à base d'ATmega32U4 comme le Romi 32U4.

La bibliothèque Servo modifiée n'interfère pas avec `Romi32U4Motors` permettant ainsi de contrôler simultanément des servos et les moteurs.

Attention: la modification décrite ici affectera toutes tous les croquis pour les contrôleurs basés sur un ATmega32U4 qui utilise la bibliothèque Servo, incluant Arduino Leonardo ou Pololu A-Star.

1. Pour commencer, il sera nécessaire de localisé le répertoire de la bibliothèque Servo d'Arduino IDE. Ensuite, ouvrir le fichier `ServoTimers.h`. Pour la version 1.6.x de l'IDE, ce fichier peut être trouvé dans `libraries/Servo/src/avr/ServoTimers.h`. Si vous utilisez Mac OS X, il sera nécessaire de faire un clic droit sur l'icône Arduino IDE et sélectionner l'option "Show Package Contents" pour voir les fichiers.
2. Ouvrir `ServoTimers.h` dans un éditeur de texte.
3. Localiser les lignes suivante dans ServoTimers.h:

```
#elif defined(__AVR_ATmega32U4__)
#define _useTimer1
typedef enum { _timer1, _Nbr_16timers } timer16_Sequence_t ;
```

1. Les deux dernières lignes indiquent que la bibliothèque utilise le Timer 1. Pour utiliser le Timer 3 à la place, changer simplement `_useTimer1` par `_useTimer3` et `_timer1` par `_timer3`.
2. Sauver le fichier.

Arduino IDE incorporera automatiquement les modification de la bibliothèque Servo. La prochaine fois que vous compilerez un croquis pour un microcontrôleur ATmega32U4, la bibliothèque servo utilisera le Timer 3 à la place du Timer 1.

AVR timers

L'ATmega32U4 dispose de 4 timers: Timer0, Timer1, Timer3 et Timer4. Chaque Timer dispose d'une ensemble de caractéristiques, tel que documenté dans la fiche technique.

- Timer0 est utilisé par l'environnement Arduino environment pour les fonctions de gestion horaire comme `millis()`.
- Timer1 est utilisé par la carte contrôleur Romi 32U4 pour contrôler les moteurs (les roues).
- Timer3 n'est pas utilisé par les bibliothèques Arduino du Romi 32U4 et peut donc être libre utilisé pour vos propres tâches/applications.
- Timer4 est utilisé par la bibliothèque Arduino Romi 32U4 pour contrôler le buzzer. La broche du Buzzer (broche numérique 6, ou PD7; sortie OC4D du Timer4) peut être libéré pour un autre usage en coupant la piste entre les deux pastilles du cavalier CMS libellé "6 = Buzzer".

Schémas et dimensions

Schémas

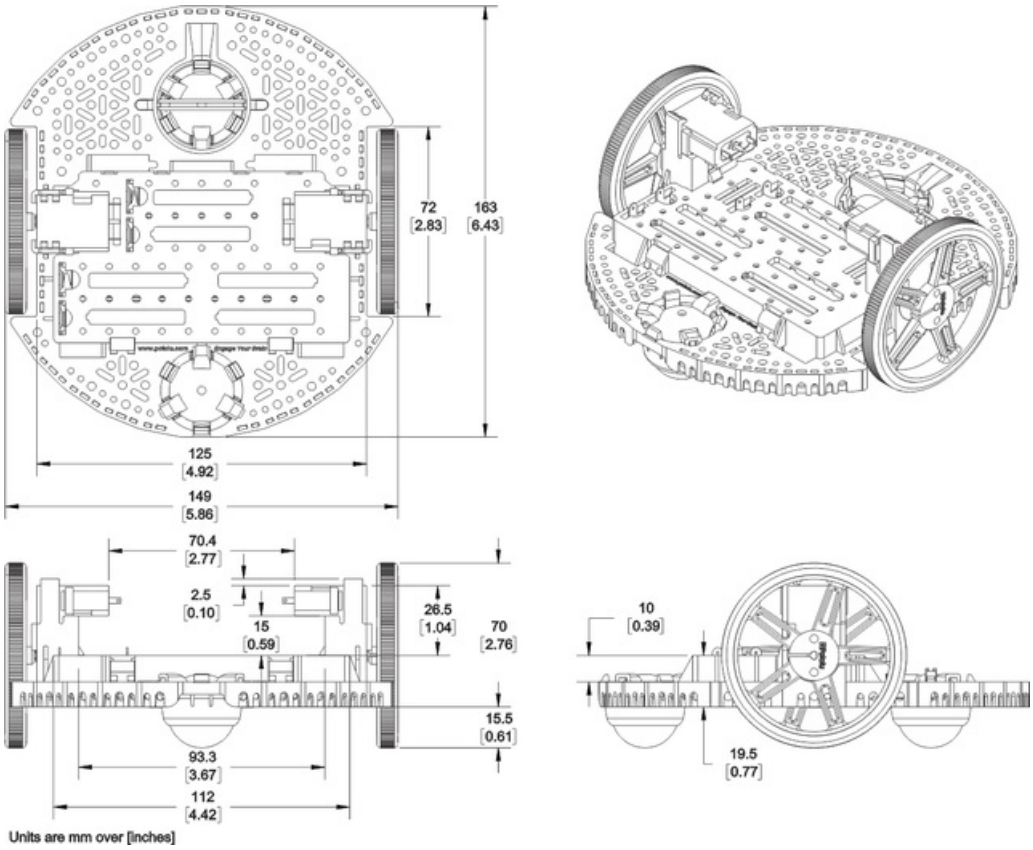
Les diagrammes de la carte de contrôle Romi 32U4 est disponible au format PDF : Schéma de la carte contrôleur Romi 32U4 https://www.pololu.com/file/0J1258/romi-32u4-control-board-schematic-diagram.pdf (*646 Kb, pdf*).

Dimensions

Un diagramme des dimensions de la carte Romi 32U4 est disponible au format PDF: Dimensions de la carte de contrôle Romi 32U4 https://www.pololu.com/file/0J1259/romi-32u4-control-board-dimensions.pdf (*604Kb, pdf*).

Les dimensions qui ne sont oncluses dans le diagramme ci-dessus peuvent être mesurées depuis le DXF suivant, qui présente le pourtour de la carte avec les tailles et position de tous les trous de la carte: Guide des perforation du Romi 32U4 https://www.pololu.com/file/0J1260/rom04a-drill.dxf (*346k, dxf*).

L'image suivante présentes les dimensions approximatives du châssis Romi https://shop.mchobby.be/category.php?id_category=141 (*lien pololu https://www.pololu.com/category/203/romi-chassis-kits*) destiné à être utilisé par la carte de contrôle Romi 32U4:



Cliquer pour agrandir

Assembler ma carte Romi 32U4

Ajouts sur la carte de contrôle

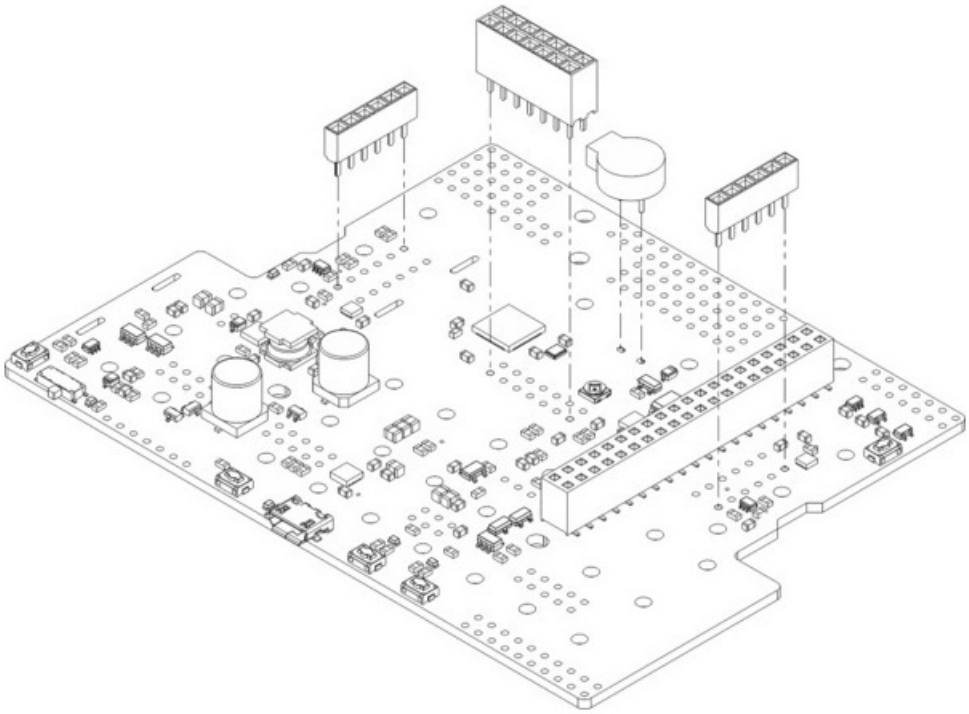
La plupart des composant de la carte de contrôle Romi 32U4 sont des composants montés en surface déjà soudé sur la carte mais il y a quelques composants traversant qu'il fait encore souder vous-même.

1. Soudez le buzzer sur le desus de la carte, faites correspondre son orientation à la sérigraphique de la carte. Une fois soudé, raccourcir les pattes (l'excédent) sous la carte.

2. Souder les deux connecteurs femelle bas profile (1×6 broches) destiné aux encodeurs. Une connecteur femelle devrait être soudé de chaque côté de la carte (dans les trous prévus à cet effet et correspondant au connecteurs mâles des encodeurs Romi (s'ils sont utilisés). Pololu recommande d'utiliser l'ensemble des trous le plus proche du bord de la carte Romi 32U4.

3. Optionel: Si vous planifiez l'usage d'un afficheur LCD (non inclus) avec la carte contrôleur alors soudez le connecteur LCD 2x7 femelle (ou 2×7 male) sur l'ensemble des des connexions portant le libellé "LCD" situé au centre de la carte. Pololu recommande d'utiliser le connecteur femelle sur la carte et des connecteurs mâle sur l'afficheur LCD (à moins d'avoir une bonne raison de faire autrement).

4. Optionel: SI vous envisagez utiliser d'autres connecteurs ou fils ALORS c'est le bon moment pour les souder.



Il est toujours possible de souder le buzzer et le connecteur LCD après le montage de la carte sur le châssis mais les souder au préalable sera plus facile et évitera de fondre accidentellement le châssis avec le fer à souder. La carte contrôleur doit être enlevée du châssis avant de pouvoir souder les connecteurs des encodeurs.

Les 4 contact de piles devraient être soudé sur la carte de contrôle **après** son montage sur le châssis, comme décrit dans les instructions d'assemblage du châssis. Il sera toujours possible d'enlever la carte (et contact de piles) du châssis après la soudure de ceux-ci.

Assembler le châssis

Une fois les composants traversants soudés sur le carte contrôleur Romi 32U4, vous pouvez suivre ces instructions du Guide utilisateur Romi Chassis de Pololu pour finir l'assemblage du châssis, montage de la carte de contrôle et soudure des contacts de piles.

Programmer la carte Romi 32U4

La carte de contrôle Romi 32U4 est conçue pour être programmée via USB à l'aide d'Arduino IDE. Il peut être programmé depuis Windows, Linux et Mac OS X. L'ATmega32U4 présent sur la carte de contrôle est livré avec un Bootloader (de la famille A-Star 32U4 / ATmega32u4) préchargé sur le microcontroleur. La sections suivante vous aidera dans vos débuts avec la carte de contrôle Romi 32U4.

Cette section aborde les 3 points suivants:

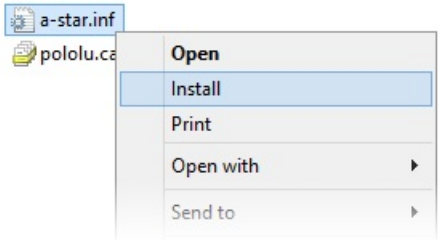
- Installer les pilotes Windows
- Programmer en utilisant Arduino IDE
- Programmer en utilisant avr-gcc et AVRDUDE

Installer les pilotes Windows

Installer les pilotes Windows

> Si vous utilisez Windows XP, il sera nécessaire d'avoir le Service Pack 3 ou Hotfix KB918365 installé avant de pouvoir installer les pilotes A-Star. Certains utilisateur ayant installer le hotfix ont rencontrés des problèmes qui auraient été résolus en faisait la mise à jour du Service Pack 3, Pololu recommande donc d'installer le Service Pack 3 au dessus de l'Hotfix.

Avant de connecter votre Pololu A-Star 32U4 (ou autre cartes de la famille 32U4) sur un ordinateur Windows, vous devriez installer ses pilotes: 1. Télécharger le pilote A-Star pour Windows https://www.pololu.com/file/0J1240/a-star-windows-1.3.0.0.zip (7k zip) et extraire le contenu du fichier ZIP dans un répertoire temporaire. (Ces pilotes sont également disponibles dans le répertoires "drivers" sur dépôt GitHub A-Star https://github.com/pololu/a-star .) 2. Ouvrir le répertoire "a-star-windows". Faire un click droit sur le fichier "a-star.inf" et selectionner "Install" (Installer).



3. Windows demandera s'il faut installer le pilote. Clicker sur "Install" (Installer... pour Windows 10, 8, 7 et Vista) ou "Continue Anyway" (continuer quand même... pour Windows XP).

4. Windows ne vous informera pas de la fin de l'installation, mais cela ne devrait prendre que quelques secondesw.

{{underline|*Utilisateurs de Windows 10, Windows 8, Windows 7 et Windows Vista:*} après avoir installé les pilotes, votre ordinateur devrait automatiquement reconnaître le périphérique lorsqu'il est connecté sur le port USB. Il n'y a pas d'autres action requises. Cependant, la première fois que vous connectez un périphérique A-Star sur votre ordinateur, Windows prendra plusieurs secondes pour détecter et le configurer correctement. La première fois que vous programmerez le périphérique, Windows prendra quelques secondes pour détecter le bootloader USB de l'A-Star et il se pourrait que l'opération de programmation pourrait échouer la première fois. De même, Windows aura besoin de refaire l'opération de détection du bootloader si la carte est connectée sur un nouveau port USB.

Utilisateur de Windows XP: Après avoir installé les pilotes, il sera nécessaire de suivre les étapes 5 à 9 pour chaque nouveau périphérique A-Star connecté sur l'ordinateur. Vous aurez besoin de suivre ces étapes la première fois que qu'un nouveau A-Star est connecté sur un nouveau port USB:

5. Connecter le périphérique sur le port USB de votre ordinateur.

6. Lorsque le Wizard "Un nouveau périphérique détecté", selectionner "Non, pas maintenant" et cliquer sur "Suivant".

7. Sur le second écran du Wizard, selectionner "Installer automatiquement le logiciel" et cliquer sur "Next".

8. Windows XP vous avertira que le pilote n'a pas été testé par Microsoft et recommande d'interrompre l'installation. Cliquez sur "Continuer de toute façon".

9. Lorsque vous avez terminé avec le Wizard, pressez simplement sur "Terminer".

Détails du port COM

Après avoir installé le pilote et branché l'A-Star, la section "Ports (COM & LPT)" du gestionnaire de périphérique devrait montrer un port COM pour le A-Star branché (il sera nommé "Pololu A-Star 32U4").

Vous pourriez voir que le port COM est nommé "USB Serial Device" dans le gestionnaire de périphérique à la place d'un nom descriptif. Cela peut arriver si vous utilisez Windows et que vous brancher un A-Star sur l'ordinateur avant d'avoir installé le pilote. Dans ce cas, Windows configurera votre connexion A-Star en utilisant le pilote série de Windows (usbser.inf) et il affichera alors "USB Serial Device" (Périphérique USB Série) comme nom du port. Le port sera toujours utilisable mais il sera difficile de dire s'il s'agit du bon port a cause du nom générique affiché dans le gestionnaire de périphérique. Pololu recommande de ficer les nom dans le gestionnaire de périphérique en faisant un clique droit sur chaque entré "USB Serial Device", puis en sélectionnant "Update Driver Software..." (mettre le pilote à jour) en puis en sélectionnant l'entrée "Search automatically for updated driver software" (chercher automatiquement pour une mise-à-jour). Windows devrait trouver le pilote déjà installé et utiliser le nom correcte pour le port.

Si vous utilisez Windows 10 (ou plus recent) et décidez de ne pas installer le pilote alors l'A-Star restera utilisable. Pour identifier le périphérique "USB Serial Device" comme l'A-Star, double-cliquez sur l'un d'entre-eux et cherchez la propriété "Hardware Ids" dans la volet "Details".

Un A-Star exécutant un croquis aura un ID `USB\VID_1FFB&PID_2300&MI_00`, sinon un A-Star en mode bootloader aura un ID `USB\VID_1FFB&PID_0101`.

Si vous voulez changer le numéro assigné au port COM de votre A-Star, vous pouvez utiliser le gestionnaire de périphérique. Double-cliquez sur le port COM pour afficher les propriétés puis cliquer sur le bouton "Avancé..." dans le volet "Paramètres".

Programmer avec Arduino IDE

Programmer en utilisant Arduino IDE

La famille des cartes à base de 32U4 peut être programmée avec le populaire environnement de programmation Arduino IDE. Arduino IDE est un environnement open-source cross-platform qui intègre un éditeur de code C++, le compilateur GNU C++ et un utilitaire de téléversement (AVRDUDE).

Pour débuter rapidement la programmation de votre périphérique avec Arduino IDE (version 1.6.4 ou suivant), en suivant les étapes suivantes:

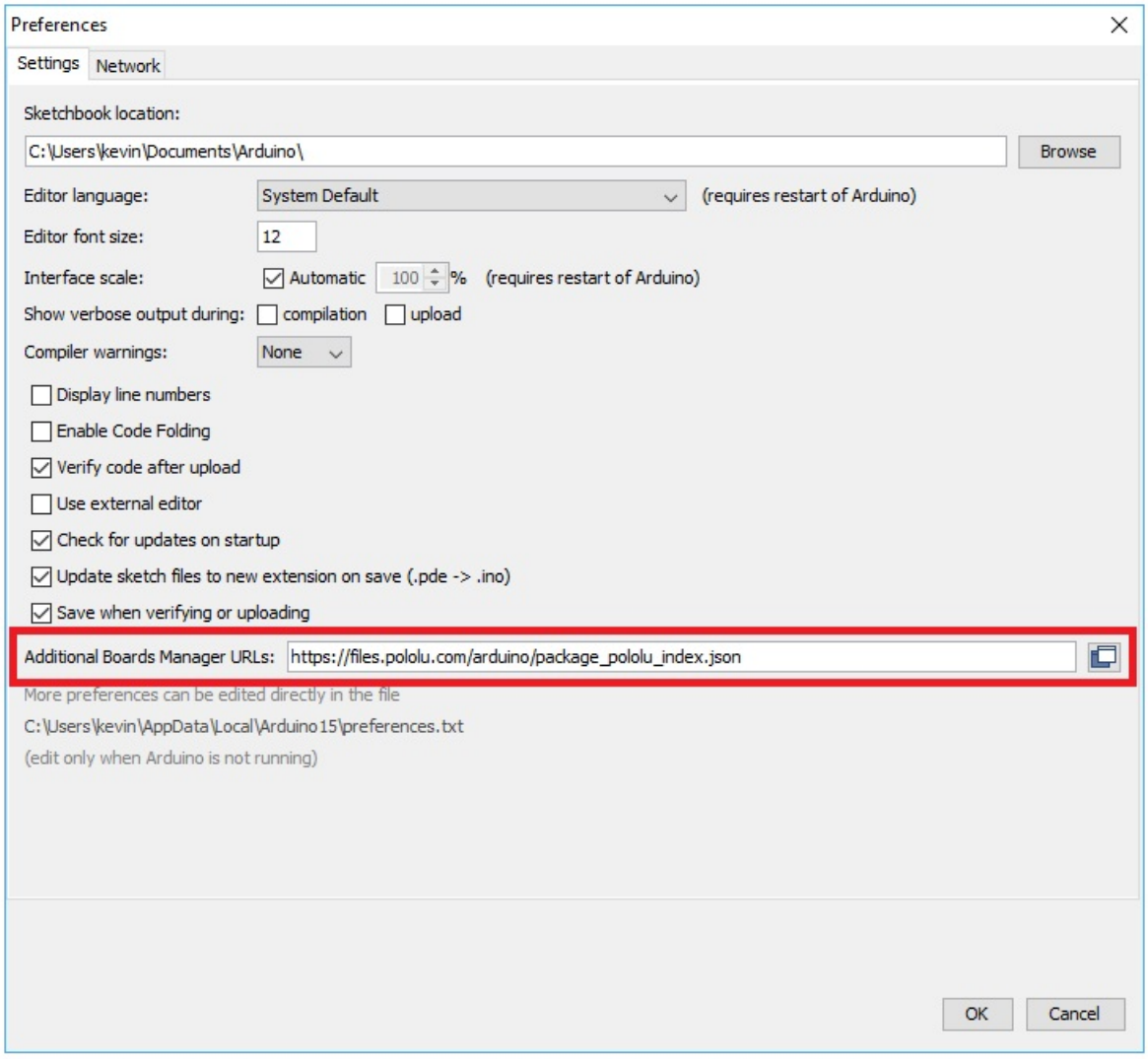
Téléchargez Arduino IDE depuis la page de télécharge d'Arduino http://arduino.cc/en/Main/Software , installez le et démarrez le.

1. Ouvrez le menu "Fichier" dans Arduino IDE (sous Windows/Linux sinon c'est le menu **Arduino** sous macOS) puis sélectionnez l'entrée "Preferences”.

2. Dans la boîte de dialogue des préférences, localisez la boîte d'édition libellée "Additional Boards Manager URLs" (URL de gestionnaire de cartes supplémentaires) comme indiqué sur la capture d'écran ci-dessous. Copiez et collez l'URL suivante dans la zone de texte:

https://files.pololu.com/arduino/package_pololu_index.json

S'il y a d'autre URLs déjà présentes alors il faut utiliser une virgule comme séparateur avant d'ajouter une nouvelle URL (cliquer sur le bouton à côté de la zone de texte permet d'ajouter l'URL sur une nouvelle ligne).



Adding a Boards Manager index for Pololu boards in the Arduino IDE's Preferences dialog.

3. Cliquez sur "OK" pour fermer la boîte de dialogue des préférences.

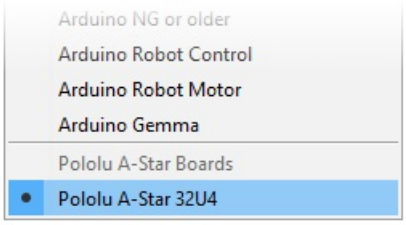
4. Dans le menu Outils > Type de carte:xx, sélectionner l'entrée "*gestionnaire de carte*" (Boards Manager) tout en haut du menu.

5. Dans le gestionnaire de carte, chercher après "Pololu A-Star Boards".

6. Sélectionnez l'entrée "Pololu A-Star Boards" dans la list puis cliquez sur le bouton "Install".

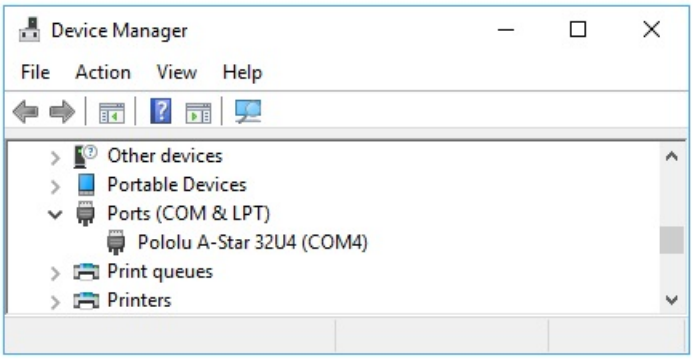
7. Une fois l'installation terminée, cliquez sur le bouton "Fermer".

8. Dans le menu Outils > Type de carte:xx, sélectionné l'entrée "Pololu A-Star 32U4". Si vous ne voyez pas le périphérique "Pololu A-Star 32U4" dans la liste alors essayez de redémarrer Arduino IDE.



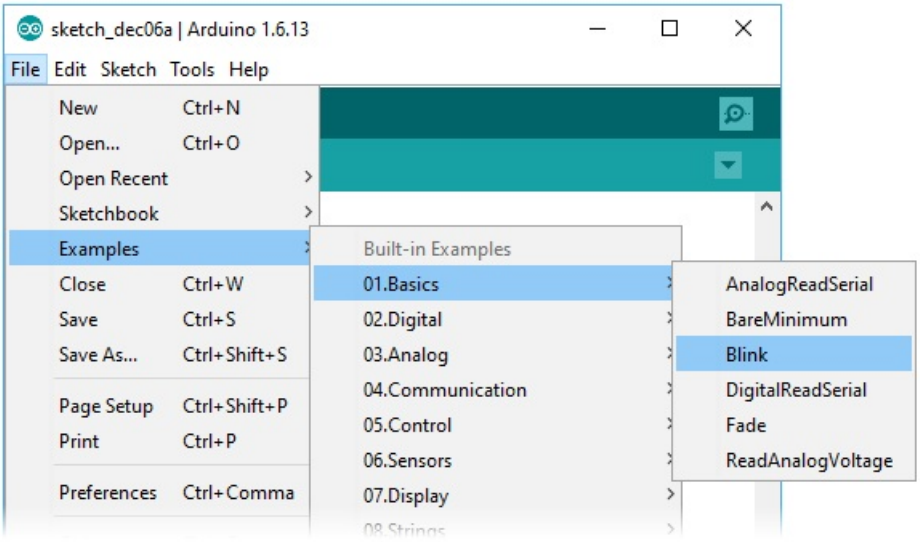
Selecting the Pololu A-Star 32U4 in the Boards menu.

9. Dans le menu **Tools > Port**, sélectionnez le port série du périphérique. Sur Windows vous pouvez déterminer le port COM du périphérique en inspectant la section “Ports (COM & LPT)” du gestionnaire de périphérique. Sous Linux, le nom du port commence par "/dev/ttyACM". Sur un Mac OS X, le nom du port début avec “/dev/tty.usbmodem”.



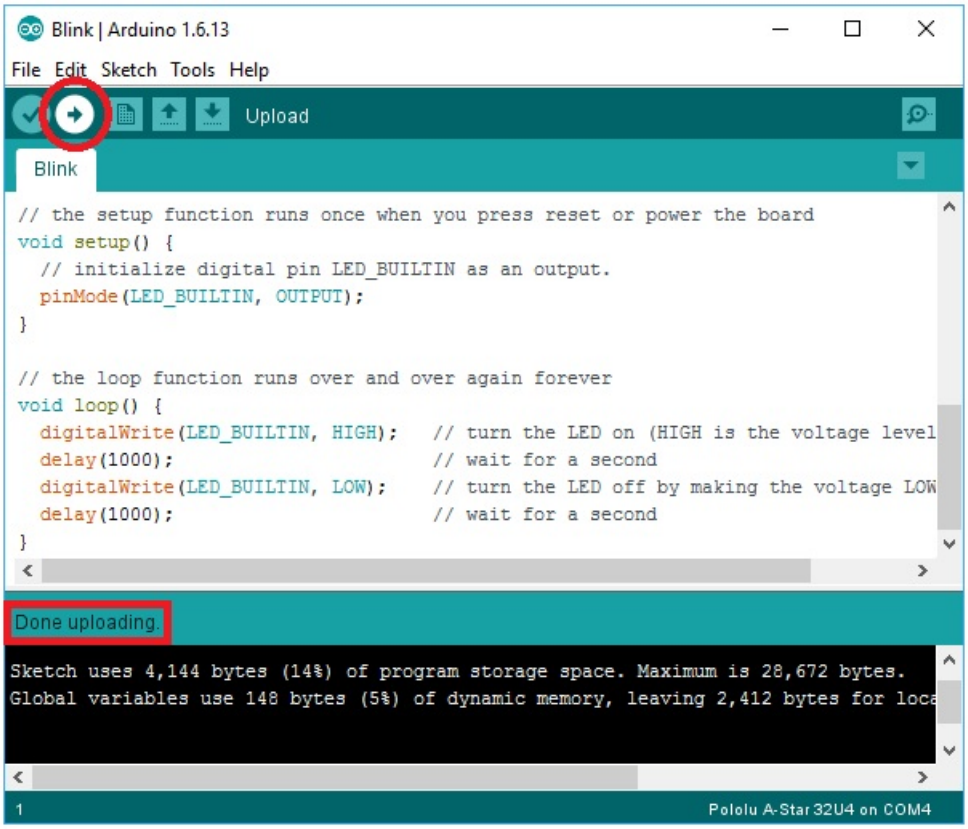
Windows 10 Device Manager showing the A-Star’s virtual COM port.

10. Ouvrez l'exemple Arduino "Blink", accessible via le menu Fichier > Exemples > 01.Basics > Blink. Le code de cet example fait clignoter la LED jaune. Lorsque l'exemple Blink est sélectionné, une nouvelle fenêtre Arduino est ouverte (vous pouvez fermer l'ancienne).



Selecting the Blink example in the Arduino IDE.

11. Pressez sur le bouton "Téléverser" (Upload) pour compiler le croquis et le téléverser sur le périphérique. Si tout se passe bien, vous verrez le message "Téléchargement terminé" apparaître dans la partie basse de la fenêtre. Si vous avez utiliser Windows et avez précédemment programmer un A-Star sur ce port USB alors Windows mettre plusieurs secondes pour détecter le bootloader de l'A-Star. Le bootloader se désactive au bout de 8 secondes et retourne à l'exécution du croquis, par conséquent, le téléversement pourrait échouer si Windows ne le reconnaît pas assez vite. Si cela arrive alors essayez encore. Si vous utiliser Windows XP et n'avez pas programmer l'A-Star sur ce port USB alors il sera nécessaire d'utiliser le Wizard "nouveau périphérique" pour pour configurer le port la première fois que vois l'utilisez. Si Arduino IDE rencontre des problèmes pour se connecter sur le port (ou l'utiliser): essayez de débrancher le périphérique, fermer les programmes qui utilise le port série, redémarrer Arduino IDE et puis rebrancher le périphérique.



Uploading a sketch to the A-Star using the Arduino IDE.

12. Si vous téléchargez le croquis Blink alors la LED jaune doit commencer à clignoter une fois toutes les deux secondes. Cependant, l'A-star est déjà libré avec ce croquis pré-programmé sur la plateforme. Changez la valeur du délai pour modifier la vitesse de clignotement et re⁻téléversez le croquis pour vous assurer que toute la proceddur fonctionne correctement.

> ***Les cartes A-Star 32U4 sont très similaires aux Arduino Leonardo pour lesquelles il n'est pas nécessaire d'installer de greffons. Si vous le voulez, vous pouvez simplement sélectionner la carte "Arduino Leonardo" dans Arduino IDE. Notez que si vous téléversez un croquis sur ce périphérique de cette façon alors la carte sera reconnue comme un Leonardo (L'entrée dans le gestionnaire de périphérique indiquera "Arduino Leonardo").***

Après avoir réussi la programmation de votre périphérique avec Arduino IDE, il y a de nombreuses ressources que vous pouvez étudier:

- Arduino IDE dispose de nombreux exemples http://arduino.cc/en/Tutorial/HomePage qui peuvent fonctionner sur A-Stars.
- Le site Arduino dispose d'une page Reference du Langage http://arduino.cc/en/Reference/HomePage , un wiki appelé Arduino Playground http://playground.arduino.cc/ et autres ressources.
- Les cartes A-Star 32U4 sont similaires à l'Arduino Leonardo et l'Arduino Micro, vous pouvez donc rechercher des projets similaire sur Internet pour les utiliser avec votre carte.

Programmer avec avr-gcc et AVRDUDE

Cette section explique comment programmer la famille des carte 32U4 de Pololu avec la chaîne de compilation avr-gcc et AVRDUDE. Cette section est destiné au utilisateurs avancés qui ne veulent pas utiliser Arduino IDE domme décrit dans la section précédente.

Prérequis

Si vous utilisez Windows, Pololu recommande de télécharger WinAVR http://winavr.sourceforge.net/ , qui contient la chaîne de compilation avr-gcc et un outil en ligne de commande appelé AVRDUDE http://www.nongnu.org/avrdude/ qui est utiisé pour téléverser le programme via le bootloader de l'A-Star. Si la version GNU Make de WinAVR crash sur votre ordinateur alors vous pouvez opter pour version la version GNU Make de Pololu https://github.com/pololu/make/releases .

Si vous utilisez Mac OS X, Pololu recommande de télécharger le CrossPack pour développement AVR http://www.obdev.at/products/crosspack .

Si vous utilisez Linux, vous aurez besoin d'installer avr-gcc, avr-libc et AVRDUDE. Les utilisateur d'Ubuntu peuvent obtenir les logiciels nécessaires à l'aide de:

```
sudo apt-get install gcc-avr avr-libc avrdude
```

Après avoir installé les pré-requis, ouvrez une invite de commande et essayez les commandes suivantes pour vous assurer que tous les utilitaires nécessaires soient bien disponibles

```
avr-gcc -v
avr-objcopy -V
make -v
avrdude
```

Si une quelconque commande échoue alors assurez vous que les exécutables nécessaires sur votre ordinateur et assurez vous que leurs répertoires soient bien mentionnés dans la variable d'environnement PATH.

Compiler un programme d'exemple

Copier le code d'exemple suivant dand un fichier nommé "main.c":

```
#define F_CPU 16000000
#include <avr/io.h>
#include <util/delay.h>

int main()
{
  DDRC |= (1 << DDC7);    // Configurer broche 13 en sortie
  while(1)
  {
    PORTC |= (1 << PORTC7);   // Allumer la LED.
    _delay_ms(500);
    PORTC &= ~(1 << PORTC7);  // Eteindre la LED.
    _delay_ms(500);
  }
}
</<syntaxhighlight>

Dans le même répertoire, créer un fichier nommé "Makefile" avec le contenu suivant:

 <syntaxhighlight>PORT=\\\\.\\USBSER000
MCU=atmega32u4
CFLAGS=-g -Wall -mcall-prologues -mmcu=$(MCU) -Os
LDFLAGS=-Wl,-gc-sections -Wl,-relax
CC=avr-gcc
TARGET=main
OBJECT_FILES=main.o

all: $(TARGET).hex

clean:
        rm -f *.o *.hex *.obj *.hex

%.hex: %.obj
        avr-objcopy -R .eeprom -O ihex $< $@

%.obj: $(OBJECT_FILES)
        $(CC) $(CFLAGS) $(OBJECT_FILES) $(LDFLAGS) -o $@

program: $(TARGET).hex
        avrdude -p $(MCU) -c avr109 -P $(PORT) -U flash:w:$(TARGET).hex
```

Vérifier que la variable PORT dans le Makefile contient le nom du port série (Virtual Com Port) sur lequel est branché le microcontrôleur. Sous Windows, `\\\\.\\USBSER000` devrait fonctionner sur l'A-Star est le seul périphérique USB connecté utilisant le pilote `usbser.sys`. Il est possible de changer cette valeur par le nom du Port COM en cours d'utilisation (ex: `COM13`).

Dans une invite de commande, naviguer vers le répertoire contenant le les fichiers Makefile et main.c. Si vous exécutez le commande `make`, le code devrait être compilé pour produire un fichier nommé "main.hex".

Programmer

Pour programmer le périphérique A-Star, il sera nécessaire de le placer en mode bootloader. Une façon d'activer le Bootloader est de réinitialiser deux fois l'AVR en 750 ms.

La plupart des cartes de la famille 32U4 (de Pololu) ont un bouton Reset qui permet de redémarrer la carte. Sur toutes les cartes de la famille 32U4, un bouton poussoir peut être connecté entre les broches RST et GND, il pourra agir comme bouton Reset (un fil fera également l'affaire).

Une fois le périphérique en mode Bootloader puis exécuter rapidement la commande `make program` pour envoyer le programme sur la carte. Si vous attendez plus de 8 secondes alors le microcontrôleur quitte le mode Bootloader et l'AVR démarre alors le programme utilisateur présent sur la carte.

Bibliothèque Arduino Romi 32U4

La carte de contrôle Romi 32U4 peut être programmé avec Arduino IDE tel que décrit dans les précédentes sections.

Afin de faciliter la programmation de tous les éléments matériels de la carte, Pololu fourni **la bibliothèque Romi32U4**. La Documentation de la bibliothèque Romi32U4 http://pololu.github.io/romi-32u4-arduino-library offre les détails concernant la bibliothèque, la bibliothèque contient plusieurs croquis.

Si vous utilisez une version 1.6.2 (ou suivante) d'Arduino IDE alors vous pouvez utiliser le gestionnaire de bibliothèque pour installer cette bibliothèque:

1. Dans Arduino IDE, ouvrir le menu "Croquis", sélectionner select "Inclure une bibliothèque", puis "Gérer les bibliothèques".
2. Chercher après "Romi32U4".
3. Cliquer sur l'entrée Romi32U4 présent dans la liste.
4. Cliquer sur "Installer".

Si cela ne fonctionne pas, vous pouvez installer la bibliothèque manuellement:

1. Télécharger et décompresser la dernière version de l'archive depuis GitHub https://github.com/pololu/romi-32u4-arduino-library .
2. Renommer le répertoire "romi-32u4-arduino-library-master" vers "Romi32U4".
3. Déplacer le répertoire "Romi32U4" dans le répertoire "libraries" présent dans le répertoire des croquis Arduino. Il est possible de repérer le répertoire des croquis en consultant les préférences d'Arduino IDE via le menu "Fichier" > "Préférence". S'il n'y a pas encode de répertoire "libraries" alors vous pouvez créer de répertoire.
4. Finalement, il sera nécessaire de redémarrer Arduino IDE après l'installation de la bibliothèque.

Après avoir installer la bibliothèque Romi32U4, vous pourrez en apprendre plus en explorant les divers croquis d'exemples inclus dans la bibliothèque et en lisant la documentation de la bibliothèque Romi32U4 http://pololu.github.io/romi-32u4-arduino-library .

Si vous utilisez le Romi 32U4 avec un Raspberry-Pi alors vous pourriez avoir envie d'utiliser la bibliothèque esclave Arduino pour Raspberry Pi https://github.com/pololu/pololu-rpi-slave-arduino-library , qui configure l'A-Star comme un esclave I2C et permet au Raspberry-Pi d'établir une communication avec l'A-Star (et de se comporter comme un maître). Voir "Interface Raspberry-Pi" pour plus d'information sur l'interface d'un Raspberry-Pi.

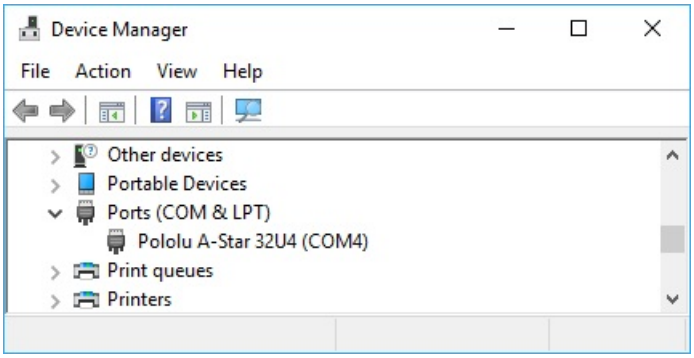
Interface USB du Romi 32U4

La famille de carte 32U4 de Pololu est basé sur le microcontrôleur AVR ATmega32U4 qui exécute le programme utilisateur et gère la connection USB avec l'ordinateur. L'AVR dispose d'un *transceiver* USB pleine vitesse. Il peut être programmé pour se présenter comme un périphérique USB auprès de l'ordinateur.

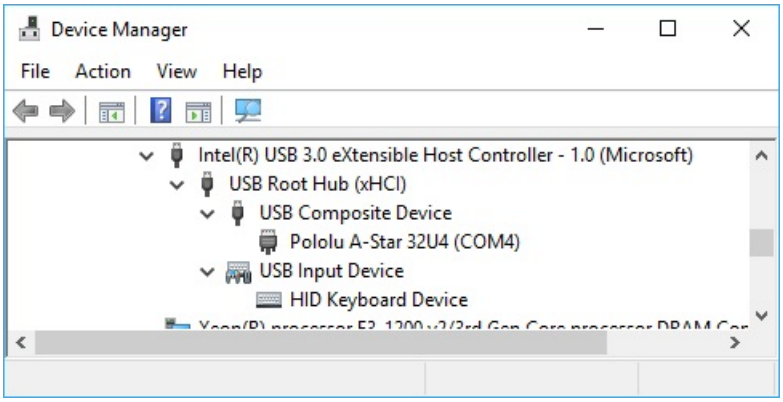
USB est un système asymétrique constitué d'un simple "hôte" connecté sur des "périphériques" multiples. L'hôte typique c'est l'ordinateur. L' ATmega32U4 peut uniquement agit en tant que périphérique USB, par conséquent le périphérique A-Star (microcontrôleur compatible Arduino) ne peut pas être connecté sur d'autres périphériques USBcomme une souris ou un clavier; il peut uniquement être connecté sur un hôte comme un ordinateur.

Programmer la carte ATmega32U4 se fait à l'aide Arduino IDE comme précédemment décrit. L'A-Star configure automatiquement son interface USB comme un périphérique USB composite avec un seul port série virtuel. Si vous programmez le microcontrôleur avec un croquis Arduino qui implémente une autre classe de périphérique USB, comme HID ou MIDI, votre ordinateur identifiera également ces périphériques enfants.

Sur un ordinateur Windows, vous pouvez identifier le port série USB en vous rendant dans le gestionnaire de périphérique (*Device Manager*) et étendre le noeuds "Ports (COM & LPT)". Vous pourrez y voir un port COM libellé "Pololu A-Star 32U4". L'identifiant du port COM (ex: COM3 ou COM4) est affiché entre parenthèse après le nom. Windows assignera un numéro de port COM différent en fonction du port USB sur lequel la carte est connectée et si celle-ci est en mode *bootloader* ou non. Si vous avez besoin, il est possible de modifier le numéro de port COM assigné à l'A-Star avec le gestionnaire de périphérique. Double-clicker sur le port COM pour ouvrir la boîte de dialogue des propriété, puis sélectionner le volet "Config. port" et le bouton "Avancé...". Depuis cette boîte de dialogue, il est possible de modifier les port COM assigné au périphérique.



Sur un ordinateur Windows, vous pouvez également voir le reste des interfaces USB depuis le Gestionnaire de Périphérique en sélectionnant `Voir > Périphérique par connexion` puis étendre les entrées jusqu'au moment où vous trouvez le port COM "Pololu A-Star 32U4". Vous trouverez, près de ce dernier les périphériques composite apparentés.



Sur un ordinateur Linux, vous pouvez voir les détails du périphérique USB en exécutant `lsusb -v -d 1ffb:` dans un terminal. Le port série virtuel peut être identifié en exécutant `ls /dev/ttyACM*` dans un terminal.

Sur un ordinateur Mac OS X, le port série virtuel est trouvé en exécutant `ls /dev/tty.usbmodem*` dans un terminal.

Vous pouvez envoyer et recevoir des octets sur le port série virtuel en utilisant un programme terminal (qui supporte le port série). Voici quelques exemples de logiciels:

- Le Moniteur Série d'Arduino IDE,
- L'utilitaire *Serial Transmitter* de Pololu https://www.pololu.com/docs/0J23 ,
- Br@y Terminal http://sites.google.com/site/terminalbpp/ ,
- PuTTY http://www.chiark.greenend.org.uk/~sgtatham/putty/ ,
- TeraTerm http://ttssh2.sourceforge.jp/ ,
- Kermit http://www.columbia.edu/kermit/ck80.html ,
- GNU Screen http://www.gnu.org/software/screen/ .

De nombreux environnements de programmation disposent également d'outils de communication via port série.

Le bootloader A-Star 32U4

Introduction

La famille de carte 32U4 de Pololu dispose d'un bootloader USB qui peut être utilisé avec Arduino IDE ou AVRDUDE pour charger un nouveau programme dans le périphérique. Cette section du document contient des détails techniques concernant le bootloader. Ces informations intéresserons les utilisateurs avancés désirants avoir une meilleure compréhension de son fonctionnement interne. Si vous voulez désirer seulement utiliser votre périphérique alors vous pouvez simplement ignorer cette section.

Le Bootloader de l'A-Star 32U4 est basé sur le bootloader Caterina https://github.com/arduino/Arduino/tree/master/hardware/arduino/avr/bootloaders/caterina utilisé par les Arduino Leonardo https://www.pololu.com/product/2192 , Arduino Micro https://www.pololu.com/product/2188 et plusieurs autres cartes ATmega32U4. Ce bootloader est *Open Source* et son code source est disponible sur GitHub https://github.com/pololu/a-star/tree/master/bootloaders/caterina . The bootloader utilise les 4 premiers kilooctets dans la mémoire programme de ATmega32U4, laissant ainsi 28 Kio pour les programmes utilisateurs. Le bootloader de l'interface USB consiste en un simple port série virtuel acceptant des commandes de programmation telles que définies dans le document AVR109 http://www.atmel.com/images/doc1644.pdf . Le bootloader démarre toujours en premier après une réinitialisation/reset de l'AVR.

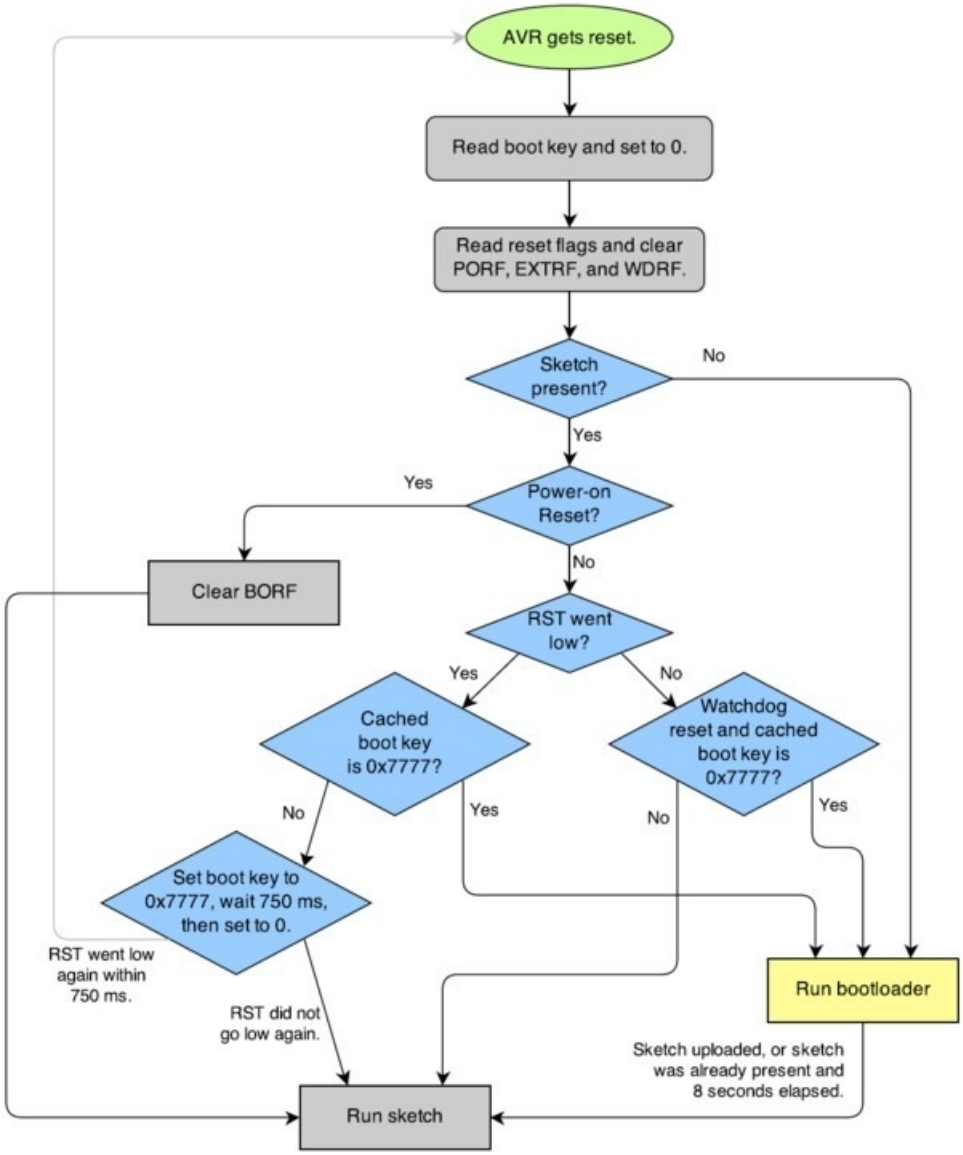
Logique de démarrage

La différence principale entre le Bootloader A-Star 32U4 et Caterina concerne la logique de démarrage. C'est la partie du bootloader qui est exécuté immédiatement après le reset de l'AVR et décide s'il doit exécuter le programme utilisateur ou le restant du Bootloader.

Le bootloader Caterina des Arduino standard est conçu pour démarrer lorsque la broche RST est placée au niveau bas. Cela signifie que is vous désirez redémarrer votre plateforme en appuyant sur le bouton Reset alors le Bootloader attendra pendant 8 secondes après le transfert d'un nouveau programme via le port USB. Au terme de ce délai (ou transmission effective d'un nouveau programme), le programme utilisateur stocké en mémoire est exécuté.

Le bootloader de A-Star 32U4 utilise une logique améliorée vous permettant d'utiliser la broche RST pour redémarrer la carte avec un faible attente. Si la broche RST passe **une fois** au niveau bas alors le programme utilisateur est démarré après 750 ms. Si la broche RST passe **une deuxième fois** au niveau bas durant cette période de 750 ms (donc Reset pressé deux fois) alors le bootloader est exécuté. Cette fonctionnalité est la même que pour la carte Micro Pro de Sparkfun.

La logique de démarrage du bootloader A-Star 32U4 est visible dans le graphique ci-dessous:



Détection Brown-out

Le "Brown out" est une brève baisse de tension qui se produit généralement/souvent durant la mise sous-tension. De nombreux microcontrôleurs intègre un mécanisme de détection de Brown-out.

Au contraire des autres cartes ATmega32U4, la famille 32U4 de Pololu inclus une détection de "brown-out". Le seuil de "brown-out" est fixé à 4.3 V et si la tension de VCC chute sous 4.3V alors l'AVR redémarrera (fera un Reset). Le bootloader à été conçu de sorte à permettre au programme utilisateur de détecter un redémarrage "brown-out".

Pour détecter le "brown-out", il faut vérifier l'état du bit BORF dans le registre MCUSR. S'il est à 1, il y a eu brown-out, vous paouvez ensuite le remettre à 0.

Voici un code d'exemple que vous pouvez utiliser dans la fonction `setup` pour détecter le "brown-out":

```
pinMode(13, OUTPUT);
if (MCUSR & (1 << BORF))
{
  // A brownout reset occurred.  Blink the LED
  // quickly for 2 seconds.
  for(uint8_t i = 0; i < 10; i++)
  {
    digitalWrite(13, HIGH);
    delay(100);
    digitalWrite(13, LOW);
    delay(100);
  }
}
MCUSR = 0;
```

Ressusciter un Romi 32U4

Pour pouvoir envoyer un nouveau programme dans votre A-Star 32U4, il faut démarrer l'A-Star 32U4 en mode bootloader et envoyer les commandes permettant sa programmation sur le port série virtuel en utilisant le logiciel approrié. Si vous programmez votre périphérique avec Arduino IDE alors le croquis téléversé dans votre carte en utilisant les commandes USB appropriée pour le placer en mode bootloader mode (et Arduino IDE envoi automatiquement les commandes lorsque vous cliquez sur le bouton de téléversement).

Cependant, si vous vous trouvez dans une situation où le périphérique ne réagit plus et que le téléversement ne fonctionne plus alors vous vous trouvez certainement dans l'une des situations suivantes:

- Vous avez accidentellement téléversé un programme non fonctionnel sur le périphérique le rendant ainsi incapable de répondre aux instructions USB. Par exemple, votre programme pourrait être bloqué dans une boucle infinie avec les interruptions désactivées.
- Vous avez téléchargés un programme qui utilise un type d'interface USB inhabituel ou pas d'interface USB (du tout).

La section suivante propose différentes procédures pour réssicuter votre périphérique.

- avec Arduino IDE
- avec AVRDUDE

avec Arduino IDE

Sommaire

Introduction

Cette section propose deux méthodes spéciales pour programmer l'A-Star (et autre cartes 32U4 de Pololu) en utilisant Arduino IDE dans le cas où votre méthode habituelle de programmation ne fonctionne pas. Ces instructions sont développées autour de Arduino IDE 1.0.5-r2 et 1.6.0 (et pourrait avoir besoin d'être mise-à-jour pour les futures versions).

Bouton Reset

Si vous disposez d'un A-Star 32U4 Micro, vous devriez connecter un bouton poussoir momentané entre les broches GND et RST (il servira de bouton RESET). D'autres cartes de la famille 32U4 disposent déjà d'un bouton Reset. Alternativement, il est toujours possible d'utiliser un bout de fil pour connecter momentanément GND et RST ensemble à la place d'utiliser un bouton Reset.

Réinitialisez la carte deux fois endéans 750 ms active le mode bootloader de la carte. Le bootloader termine automatiquement son exécution au bout de 8 secondes et essaye de démarrer le programme utilisateur (si aucun croquis est téléversé). Pour réactiver la carte, il faut téléverser un croquis endéans les 8 secondes. Le croquis "Blink without delay" est un excellent candidat.

Dans le modue bootloader, la LED jaune pulse (celle portant le libellé LED 13). En surveillant la LED il est facile d'identifier le mode de fonctionnement du microcontroleur. De même, Pololu recommande d'activer l'affichage des messages détaillés durant le téléversement du croquis (voir l'option "Fichiers > Préférences"). En surveillant la LED et les informations détaillées durant la procédure vous aidera à comprendre ce qui se passe.

La méthode uploading-before-bootloader

Le but de la méthode *uploading-before-bootloader* (téléverser avec le bootloader) consiste en la sectionner un port série non-existant dans Arduino IDE et ensuite de s'assurer que Arduino IDE entre en phase de téléversement AVANT que le bootloader soit activé sur le microcontroleur. Cette méthode a été testée sur Arduino 1.0.5-r2 et 1.6.0. Cette méthode ne fonctionne pas avec Arduino 1.5.6-r2 car cette version de l'IDE provoque un message d'erreur fatale si le port série n'est pas sélectionné au début de la phase de téléversement (ex: "Carte sur COM7 n'est pas disponible").

1. Connectez le périphérique sur votre ordinateur via USB.
2. Dans le menu "Outils", ouvrir le sous-menu "Carte" et sélectionnez "Pololu A-Star 32U4".
3. Dans le menu "Outils", ouvrir le sous-menu "Port" et vérifier qu'il n'y a pas de port sélectionné. Si le menu "Port" est grisé ou qu'aucun port est sélectionné alors tout est en bonne ordre et vous pouvez passer à l'étape 6.
4. Réinitialisez la carte deux fois pour activer le mode bootloader. Pendant que la carte est en mode bootloader, sélectionnez rapidement le nouveau port série qui correspond à la carte en mode bootloader (par l'intermédiaire du menu "Port").
5. Après 8 secondes, le bootloader cesse de fonctionner et essaye d'exécuter le programme utilisateur une nouvelle fois. Attendre que le bootloader termine son exécution et vérifier que le monu port soit grisé ou qu'aucun port n'y soit soit sélectionné.
6. Cliquer sur le bouton téléverser. Arduino IDE commencera à compiler le croquis puis démarre le téléversement.
7. Dès que la barre de statut d'Arduino IDE affiche "Téléversement..." (ou Uploading en anglais), pressez deux fois le bouton RESET pour activer le mode bootloader.

Arduino IDE restera en mode téléversement pendant 10 secondes, attendant qu'un nouveau port série apparaisse. Une fois le port série du bootloader apparu, Arduino IDE s'y connectera et enverra les commandes de programmation.

La méthode bootloader-before-uploading

Le but de la méthode "bootloader-before-uploading" (bootloader avant téléversement) consiste à sélectionner le port virtuel du bootloader dans Arduino IDE et de s'assurer que la carte est en mode bootloader au moment où Arduino IDE débute sa phase de téléversement.

1. Connectez le périphérique sur votre ordinateur via USB.
2. Dans le menu "Outils", ouvrez le sous-menu "Cartes" et vérifier si l'entrée "Pololu A-Star 32U4 (bootloader port)" est disponible. Si cette entrée est visible, vous pouvez sauter directement à l'étape 6.
3. Si vous utilisez la version 1.0.x d'Arduino IDE, ouvrez le fichier **[emplacement des croquis]/hardware/pololu/boards.txt** en utilisant un éditeur de texte. Si vous utilisez la version 1.5.x d'Arduino IDE, ouvrez le fichier **[emplacement des croquis]/hardware/pololu/avr/boards.txt** en utilisant un éditeur de texte. Vous pouvez identifier l'emplacement des croquis depuis la boîte de dialogue des préférences d'Arduino IDE. Le fichier que vous recherchez est le greffons dédié aux cartes A-Star.
4. Dans le fichier boards.txt que vous avez ouvert, cherchez les lignes en bas des fichier commençant par `#a-star32U4bp`. Décommentez chacune de ces lignes en effacant le caractère “**#**” cen début de ligne puis sauvez le fichier.
5. Fermez Arduino IDE puis redémarrez le.
6. Dans le menu "Outils", ouvrez le sous-menu "Cartes" et sélectionnez "Pololu A-Star 32U4 (bootloader port)". Cette entrée étant configurée de sorte qu'Arduino IDE enverra les commande de programmation directement sur le port série sélectionné (plutôt que d'essayer d'envoyer des commandes USB spécifiques pour passer le port en mode Bootloader et attendre qu'un nouveau port USB apparaisse sur le système). En sélectionnant cette entrée, le flux des opérations pour la programmation devient plus facile... surtout sous Windows.
7. Préparez votre ordinateur pour afficher la liste des ports séries virtuels. Si vous utilisez Windows, cela signifie que vous devez ouvrir le gestionnaire de périphériques. Si vous utilisez Linux ou Mac OS X alors il faudra ouvrir un terminal et saisir la commande `ls /dev/tty*` mais pressez la touche Return/Retour clavier qu'une fois la carte en mode Bootloader (voir les étapes suivantes).
8. Pressez deux fois la bouton Reset de la carte pour la passer en mode Bootloader. Alors que la carte est en mode Bootloader (pendant 8 secondes), regardez rapidement la liste des port série disponibles sur votre système d'exploitation afin de savoir à quel port le bootloader est associé.
9. Pressez le bouton Reset deux fois (un nouvelle fois) pour remettre la carte en mode Bootloader (encore). Tandis que la carte est en mode Bootloader, sélectionnez rapidement le port série correspondant dans Arduino IDE. Le port peut être sélectionné dans le sous-menu "Port" du menu "Outils".
10. Dans Arduino IDE, pressez sur le bouton "Verify" pour compilez votre croquis. Cela vous aidera à anticiper le temps nécessaire à la compilation de votre croquis pour l'étape suivant.
11. Pressez ke bouton Reset deux fois pour placer la carte en mode bootloader (encore une fois). Et aussi tôt que la LED jaune pulse sur la carte, pressez sur le bouton Téléversez/Upload.

Arduino IDE compilera le croquis et le téléversea sur le port série sélectionné.

Si la compilation du croquis prends plus de 8 secondes alors la procédure échouera parce que la carte aura quitté le mode Bootloader au moment du Téléversement (et la carte essayera alors de démarrer le croquis défaillant déjà présent en mémoire. Si cela arrive, alors essayez cette procédure avec un croquis beaucoup plus simple et plus rapide à compiler comme l'exemple "Blink" que vous pourrez charger via le menu **Fichier > Exemple > 01.Basics > Blink** .

Après la résurrection de votre périphérique, assurez-vous que la configuration de la carte soit bien repositionner sur "Pololu A-Star 32U4" et sélectionnez le bon Port.

avec AVRDUDE

Cette section explique une méthode spéciale permettant de réssuciter un A-Star (et autres cartes de la famille 32U4) en utilisant un utilitaire en ligne de commande AVRDUDE http://www.nongnu.org/avrdude/ dans le cas où la méthode de programmation habituelle échoue. AVRDUDE est un accronyme signifiant "AVR Downloader/UploaDEr" (Téléchargement / Téléversement pour AVR) et il est compatible avec le bootloader de l'A-Star.

Si vous disposez d'un A-Star 32U4 Micro, vous devriez connecter un bouton poussoir entre les broches RST et GND pour facilier les manipulations nécessaires (il servira de bouton Reset). Les autres carte de la famille 32U4 produite pas Pololu dispose d'un bouton Reset sur la carte.

1. Connectez le périphérique sur votre ordinateur via la connexion USB.
2. Préparer l'ordinateur pour afficher la liste des périphériques et leur port COM virtuel. Si vous utilisez Windows alors il faut ouvrir le gestionnaire de périphérique. Si vous êtes sur Linus ou Mac OS X alors ouvrez un terminal et saisissez la commande `ls /dev/tty*` mais ne pressez pas encore la touche de retour clavier (voir point suivant).
3. Pressez le bouton Reset deux fois (endéans les 750 ms) pour mettre l'AVR en mode bootloader. Vous devriez vois la LED jaune de la carte pulser indiquant que la carte est en mode Bootloader. Alors que la carte est en mode Bootloader, cherchez rapidement après le nouveau port série apparu dans la le gestionnaire de périphérique (ou presser le retour clavier dans le terminal pour lister les périphériques). Cela permettra d'identifier la carte le port série associé au bootloader de la carte.
4. Dans la commande suivante saisie dans votre terminal, remplacez le COM4 avec le port série du Bootloader que vous avez identifié grâce aux points précédents... mais ne pressez pas encore la retour clavier. Cette commande effacera le programme non fonctionnel de la mémoire flash du périphérique tout en préservant le bootloader.
   `avrdude -c avr109 -p atmega32U4 -P COM4 -e`
5. Pressez le bouton Reset deux fois (endéans les 750 ms) pour placer l'AVR en mode Bootloader.
6. Exécutez la commande saisie précédemment. La commande à besoin d'être exécutée endéans les 8 secondes d'exécution du Bootloader (ou sinon, le microcontrôleur essayera à nouveau d'exécuter le croquis défaillant présent dans la flash du MicroContrôleur).

En suivant les instructions ci-dessus, le croquis défaillant présent sur le périphérique sera effacé et seul le bootloader restera présent dans dans la mémoire flash. Le périphérique reste continuellement en mode Bootloader. Maintenant, vous pourrez programmer la carte via Arduino IDE ou AVRDUDE dans aucun problème.

Autres ressources

Si vous désirez en apprendre plus sur la carte de contrôle Romi 32U4, vous pouvez consulter les ressources suivantes:

- La page Arduino IDE dispose de nombreux exemples http://arduino.cc/en/Tutorial/HomePage fonctionnant sur la carte de contrôle Romi 32U4 (notez quand même que la carte de contrôle pourrait rencontrer des conflits matériels avec certains exemples).
- Le site Arduino dispose d'une Reference de Langage http://arduino.cc/en/Reference/HomePage , un wiki communautaire nommé Arduino Playground http://playground.arduino.cc/ et autres ressources.
- La carte de contrôle Romi 32U4 utilise le même microcontrôleur qu'un Arduino Leonardo https://www.pololu.com/product/2192 et Arduino Micro https://www.pololu.com/product/2188 . Vous pouvez donc trouver des codes d'exemples et ressources pratiques sur Internet pour ces cartes.
- Documentation Atmel ATmega32U4 https://www.microchip.com/wwwproducts/en/ATmega32u4 inclus une fiche technique et nombreux documents.
- Page d'accueil AVR Libc http://www.nongnu.org/avr-libc/ : cette page documente la bibliothèque des fonctions standards qui peuvent être utilisées avec les compilateurs GNU C et C++ pour AVR.
- La bibliothèque Arduino standard pour le Romi 32U4 https://github.com/pololu/romi-32u4-arduino-library
- La bibliothèque Romi 32U4 http://pololu.github.io/romi-32u4-arduino-library
- LUFA (*Lightweight USB Framework for AVRs*) http://www.fourwalledcubicle.com/LUFA.php
- WinAVR http://winavr.sourceforge.net/
- Atmel Studio 7 https://www.microchip.com/avr-support/atmel-studio-7
- AVRDUDE http://www.nongnu.org/avrdude/
- AVR Freaks http://www.avrfreaks.net/

Les fiches techniques pour les composants utilisés sur la carte de contrôle de la carte Romi 32U4:

- Documentation ATmega32U4 https://www.microchip.com/wwwproducts/en/ATmega32u4
- Fiche technique du pilote moteur DRV8838 de Texas Instruments https://www.pololu.com/file/0J806/drv8838.pdf.redirect (1k redirect)
- Fiche technique de l'accéléromètre LSM6DS33 3D de ST et du module Gyroscope 3D https://www.pololu.com/file/0J1087/LSM6DS33.pdf (1MB pdf)
- Multiplexeur d'alimentation TPS2113A de Texas Instruments https://www.pololu.com/file/0J771/tps2113a.pdf.redirect (1k redirect)

Pour finir, vous pouvez poster vos commentaires et questions sur le Forum de Pololu Robotics https://forum.pololu.com/ (en anglais *of course*)!
