contrôleur de vitesse par GPS

plus de contraventions pour excès de vitesse

Olivier Croiset (France)

Il existe de nombreux exemples sur l'internet montrant comment utiliser un module GPS dans un projet de microcontrôleur. Mais que faire des coordonnées GPS, si ce n'est les reporter sur une carte ? J'ai voulu faire quelque chose de plus utile et j'ai créé ce contrôleur de vitesse basé sur le GPS. Ce dispositif peut être utilisé dans n'importe quel véhicule, aussi bien une voiture qu'un bateau. Il vous permet de conduire sans avoir les yeux rivés sur le compteur de vitesse et d'éviter les mauvaises surprises par la suite dans le courrier.

L'appareil présenté ici n'est pas un limiteur de vitesse, qui nécessiterait une modification du véhicule, mais une alarme qui signale qu'une vitesse présélectionnée est atteinte (ou dépassée). Pour que vous ne quittiez pas la route des yeux, l'avertisseur émet deux longs bips si la vitesse sélectionnée est dépassée, tandis que deux bips courts indiquent que le véhicule est repassé sous la limite.

Quatre consignes de limites de vitesse

Le contrôleur de vitesse dispose de quatre valeurs programmables de limites de vitesse, ce qui devrait suffire pour les limites rencontrées sur la plupart des trajets. De toute façon, l'écran ne permet guère d'en afficher plus de quatre. Pendant la conduite, le conducteur doit être en mesure de choisir rapidement l'une des valeurs, si possible sans regarder l'appareil. Je n'ai pas voulu utiliser d'écran tactile ; le conducteur doit sentir physiquement les pressions exercées sur les boutons. Pour des raisons ergonomiques, les boutons sont placés au-dessus des quatre limites de vitesse affichées à l'écran.

Une bonne raison pour s'essayer au 32 bits

Mes premiers essais utilisant un microcontrôleur 8 bits ATmega328 associé à un module GPS m'ont rapidement convaincu que ses 32 ko de mémoire flash ne suffiraient pas pour une application plus évoluée, surtout avec un affichage graphique des données sur un écran TFT.
Par conséquent, après avoir épuisé les nombreuses qualités de l'ATmega328, j'ai décidé de passer au niveau au-dessus et d'essayer un microcontrôleur 32 bits STM32. Sa forme la plus adaptée aux bricoleurs de mon espèce est la carte BluePill avec son STM32F103C8. Cette petite carte facilite considérablement l'utilisation de ce microcontrôleur. Elle est facile à programmer via son port USB à l'aide de l'EDI Arduino (à condition que la carte soit programmée avec un chargeur adéquat au préalable), elle est compacte et ne coûte que quelques euros.

Tailles de la mémoire flash

Ce projet est une première tentative d'utilisation du STM32F103C8. Notez que, officiellement, le ***C8*** est équipé de 64 ko de mémoire flash, alors que la version ***CB*** en a 128 ko, mais il arrive que le C8 soit lui aussi équipé de 128 Ko (pièces de contrefaçon ?). Cela permettra, pour ceux qui seraient tentés, de poursuivre l'apprentissage en ajoutant d'autres périphériques, tels qu'une carte SD, un module SIM, la transmission de données par liaison radio, etc.

Le module GPS

Le module GPS utilisé dans ce projet est un module NEO-6 de u-blox. Son utilisation est facile, car la bibliothèque TinyGPS+ [2] se charge de décoder le flux de données au format NMEA 0183 émis par le module GPS. Les principaux paramètres que nous obtenons de cette manière sont les suivants :

- Date et heure UTC ;
- Longitude et latitude, en degrés décimaux ;
- Vitesse ;
- Direction (cap) ;
- Altitude ;
- Le nombre de satellites visibles ;
- La valeur de la précision horizontale (HDOP).

La précision horizontale, ou HDOP (horizontal dilution of precision) [2], dépend de la position des satellites à portée du récepteur GPS. Plus cette valeur est faible (proche de 1), meilleure est la précision des coordonnées. Une valeur HDOP de 10 indique que les coordonnées sont peu précises, voire invalides. Le contrôleur de vitesse n'utilise pas ce paramètre.

L'écran TFT

Pour l'affichage, j'ai utilisé un module d'affichage TFT standard de 2,2 pouces avec un pilote ILI9341 et une interface SPI. Il a une résolution de 320 × 240 pixels, ce qui est suffisant pour les données que nous voulons afficher (quatre limites de vitesse avec un bref message indiquant si la limite de vitesse a été atteinte ou non). Le contrôleur de vitesse dispose de deux modes d'affichage principaux en utilisation normale. Le mode d'affichage en cours est sauvegardé dans l'EEPROM et est restauré lorsque le système est remis en marche. Les préréglages des limites de vitesse sont également stockés dans l'EEPROM. Seuls six octets de cette EEPROM sont utilisés : quatre octets pour les limites de vitesse, un octet pour le dernier écran sélectionné et un octet pour la dernière limite de vitesse sélectionnée.

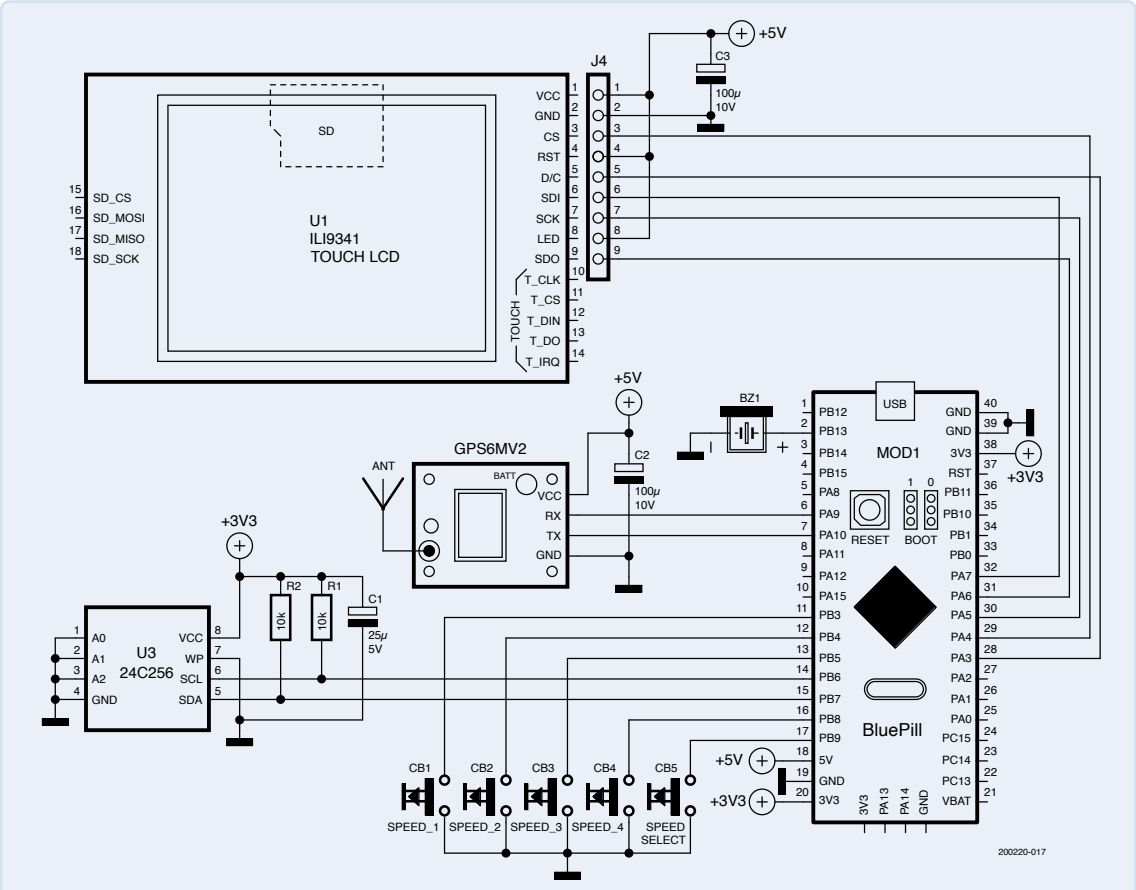

Figure 1. Le contrôleur de vitesse combine des modules facilement disponibles avec quelques composants électroniques.

Le schéma du circuit

Le schéma du contrôleur de vitesse est présenté à la **figure 1**. Le module BluePill dans le coin inférieur droit est le cœur du circuit.

L'alimentation électrique se fait par le biais du connecteur USB de la BluePill. Les véhicules récents sont souvent équipés d'un connecteur USB ; dans le cas contraire, un adaptateur 12 V vers USB est nécessaire (ou une batterie powerbank). Le module BluePill est alimenté par une tension de 5 V ; il possède son propre régulateur de 3,3 V, qui alimente l'EEPROM. L'écran TFT et le module GPS sont alimentés en 5 V. La consommation totale du circuit est d'environ 200 mA.

L'écran est contrôlé par un bus SPI, le module GPS est contrôlé par une liaison série et l'EEPROM est connectée au bus I²C (avec ses deux résistances de rappel). Pour mon prototype, j'ai utilisé une EEPROM 24C256 de 32 Ko, mais une 24C01 de 128 octets serait déjà largement suffisante.

Douze ports GPIO restent libres. Il est donc possible d'ajouter des périphériques, des éléments et des fonctions supplémentaires, tant que le programme d'application tient dans la mémoire flash. Le microprogramme compilé de ce projet occupe 64 020 octets, soit la quasi-totalité de la mémoire flash.

Construction d'un contrôleur de vitesse

Le critère le plus important à prendre en compte dans la réalisation de ce projet est l'ergonomie. Le conducteur doit pouvoir trouver les boutons sans se tromper. Les boutons-poussoirs ont une tige de 25 mm – la plus longue que j'ai pu trouver.

J'ai réalisé ce prototype sur une carte de prototypage (**figure 2**). Un circuit imprimé (PCB) permettrait une réalisation plus compacte dans un boîtier plus fin. Le connecteur USB doit rester accessible, car il sert à alimenter le circuit en utilisation normale. Il est également utilisé pour mettre à jour le logiciel.

L'antenne GPS doit être aussi éloignée que possible du connecteur de l'écran TFT (au moins quelques centimètres).

Notes sur le module GPS

Le module GPS est doté d'une minuscule batterie de sauvegarde des données. Si le GPS n'est pas utilisé pendant plusieurs jours d'affilée, la batterie se décharge et les données sont perdues. Lorsque vous rallumez le GPS, il se peut que vous deviez attendre quelques minutes pour restaurer les données (et recharger la batterie). La LED du module GPS clignote lorsqu'il réussit à décoder les données satellitaires. Quelques instants plus tard, le nombre de satellites s'affiche.

L'antenne GPS doit « voir » les satellites. La réception des signaux doit donc être entravée le moins possible. En zone urbaine, entre deux bâtiments, la réception peut être difficile. Néanmoins, le contrôleur de vitesse placé au niveau des pieds du passager dans ma voiture fonctionne correctement.

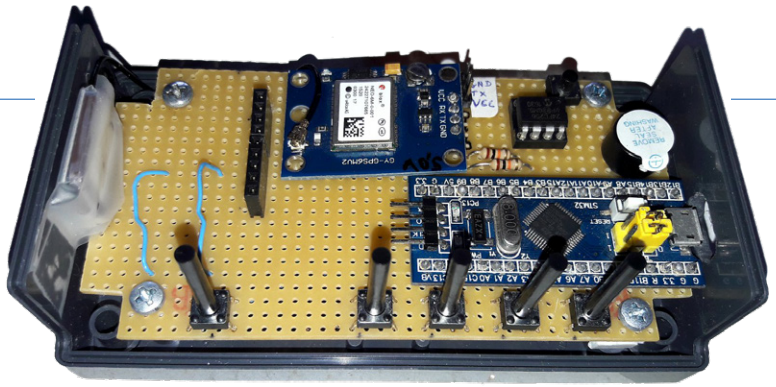

Figure 2. Le prototype a été construit sur une carte de prototypage. Les dimensions du boîtier sont de 12 x 6,5 x 4 cm.

Réglage des limites de vitesse

Cette opération ne doit être effectuée que lorsque le véhicule est à l'arrêt ! Lors de la première mise sous tension du contrôleur de vitesse, les limites de vitesse sont toutes réglées sur 255. Cela correspond à FF en hexadécimal, ce que contient une EEPROM vierge.

Dans l'écran 3 (**figure 5**), qui affiche les données des satellites, appuyez sur le troisième bouton pour activer le réglage de la limite de vitesse. Ensuite, utilisez le premier et le deuxième bouton pour régler la limite souhaitée. Appuyez à nouveau sur le troisième bouton pour valider la limite de vitesse et passer à la suivante, et ainsi de suite. Veillez à ce que les limites soient dans un ordre croissant, car le logiciel ne les trie pas.

Figure 3. L'écran 1 montre les limitations de vitesse sous la forme de panneaux de signalisation ronds accompagnés d'un bref message.

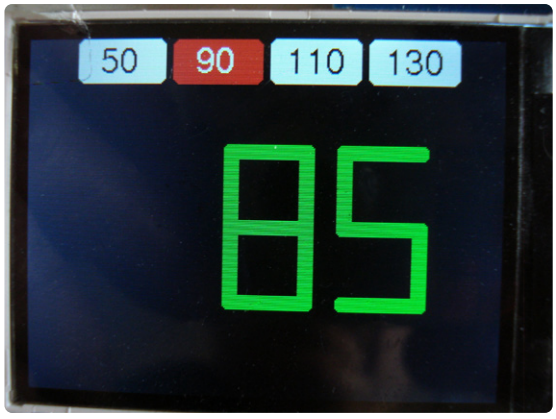

Figure 4. Sur l'écran 2, la vitesse réelle est affichée en chiffres de style 7 segments.

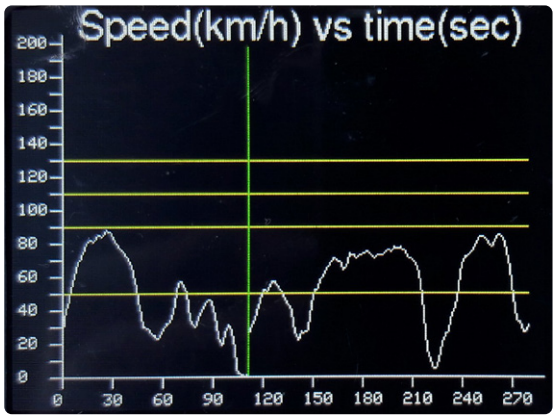

Figure 5. L'écran 5 montre la courbe de l'historique de la vitesse. Les quatre lignes horizontales jaunes indiquent les quatre limites programmées et la ligne verticale verte indique la vitesse actuelle.

Figure 6. Les données GPS sont disponibles sur l'écran 3. C'est également par cet écran qu'on accède à l'écran 4 de programmation des valeurs des limites de vitesse.

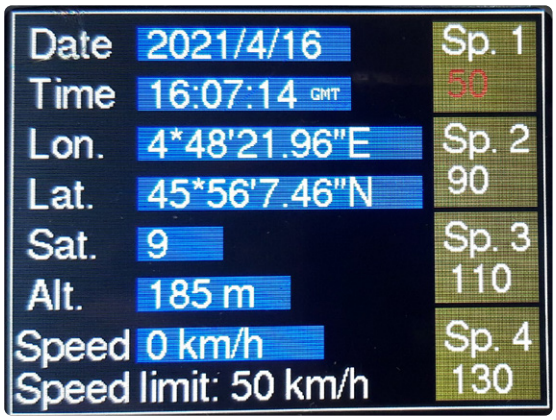

Figure 7. L'écran 4 permet de programmer les limites de vitesse. Ici, la première limite, « Sp. 1 », est en cours de réglage (en rouge, si vous regardez bien).

Utilisation du contrôleur de vitesse

Le contrôleur de vitesse est facile à utiliser. Toutefois, comme il est censé être utilisé dans un véhicule en mouvement, veillez à ce que l'appareil et surtout son câble d'alimentation ne soient pas gênants pendant la conduite. Une fois l'appareil placé à un endroit approprié, sélectionnez le mode d'affichage souhaité. L'écran 1 (**figure 3**) affiche des panneaux de signalisation ronds, l'écran 2 affiche la vitesse actuelle sous forme de chiffres à 7 segments (**figure 4**). Sélectionnez ensuite la limite de vitesse que vous souhaitez utiliser. L'avertisseur sonore retentit si vous roulez trop vite.

Votre passager peut visualiser la courbe de vitesse sur une période de 4,5 minutes (270 secondes) en sélectionnant l'écran 5 (**figure 5**). Sur cet écran, l'avertisseur sonore ne retentit pas. Les lignes horizontales indiquent les quatre limitations de vitesse. La ligne verticale «maintenant» indique la vitesse actuelle. Un graphique défilant aurait été plus joli, mais il nécessite une actualisation complète pour chaque nouveau point de données, ce qui est lent. Il est beaucoup plus facile de déplacer le curseur.

Écran des données GPS

L'écran 3 (**figure 6**) affiche les données GPS et vous permet de vous situer sur une carte à l'aide de marqueurs de coordonnées. La précision est d'environ 30 mètres, ce qui est tout à fait satisfaisant pour notre petit appareil bon marché. Dans ce mode d'affichage, l'avertisseur reste muet. L'écran 4 (**figure 7**) est identique à l'écran 3, sauf qu'il permet de programmer les limitations de vitesse. Notez que la date et l'heure du GPS sont des valeurs UTC, c'est-à-dire celles du méridien de Greenwich. Si vous souhaitez les utiliser, vous devez corriger ces valeurs en fonction de votre fuseau horaire local et de l'heure d'été ou d'hiver. N'étant pas une horloge, le contrôleur de vitesse n'inclut pas de menu de gestion de l'écart.

Tableau 1. Les différents écrans et comment naviguer.

Écran	Bouton 1	Bouton 2	Bouton 3	Bouton 4	Bouton 5
Ecran 1 (Panneaux ronds)	Limite de vitesse 1	Limite de vitesse 2	Limite de vitesse 3	Limite de vitesse 4	Aller à l'écran 2
Ecran 2 (Affichage 7 segments)	Limite de vitesse 1	Limite de vitesse 2	Limite de vitesse 3	Limite de vitesse 4	Aller à l'écran 3
Ecran 3 (Données GPS)	Limite de vitesse 1	Limite de vitesse 2	Aller à l'écran 4	Pas d'action	Aller à l'écran 1
Ecran 4 (Programmer les limites)	Diminuer la limite (-5 km/h)	Augmenter la limite (+5 km/h)	Valider et passer à la limite suivante, Retourner à l'écran 3 une fois terminé	Aller à l'écran 5	Pas d'action
Écran 5 (Courbe de vitesse)	Pas d'action	Pas d'action	Pas d'action	Aller à l'écran 2	Pas d'action

Logiciel

Le programme est facile à modifier et le code source peut être téléchargé à partir de [3]. Il s'agit d'un croquis Arduino, qui nécessite le STM32 Boards Package for Arduino (nous avons utilisé la version 2.3.0) [4]. La majeure partie du code est consacrée à l'interface utilisateur graphique, le reste se charge de la tâche simple consistant à recevoir les données GPS, à extraire la vitesse et à la comparer à une limite.

Comme d'habitude dans un sketch Arduino, la fonction `setup()` se charge d'initialiser tous les périphériques. Une fois cette opération terminée, elle affiche un écran de bienvenue pendant cinq secondes.

La fonction `loop()` commence par lire les boutons-poussoirs avant de vérifier si le GPS contient des données fraîches (dans la fonction `dataDecode()`). Ensuite, en fonction de l'écran actif, les données correspondantes sont affichées. Si la vitesse actuelle est supérieure à la limite de vitesse active, une alarme retentit.

Notez que des informations de débogage et d'état sont envoyées vers le port série (115200,N,8,1).

Pour aller plus loin

Voici quelques éléments que vous pouvez ajouter ou modifier :

- Utiliser d'autres unités de vitesse, pour un bateau, par exemple. La bibliothèque GPS vous permet d'utiliser d'autres unités de vitesse, telles que les nœuds ou les miles/heure (MPH). Il faudra pour cela modifier les écrans (par exemple, remplacer les km/h par des nœuds) et la taille des incréments de limite de vitesse (dans la fonction `SpeedSettings()`).

Si votre module BluePill dispose de 128 Ko de mémoire flash, des fonctions supplémentaires peuvent être ajoutées, telles que :

- Enregistrement sur carte SD. Au dos du module d'affichage TFT se trouve un emplacement pour carte SD. Il peut être utilisé pour enregistrer, par exemple, les données GPS afin de les consulter sur un PC.
- Ajouter un écran tactile pour de nouvelles fonctions. Dans ce cas, veillez à la sécurité de l'utilisateur.
- Réglage de l'heure locale.

Vf : Helmut Müller — 200220-04

Des questions, des commentaires ?

Si vous avez des questions techniques, n'hésitez pas à envoyer un courriel à l'équipe éditoriale d'Elektor à l'adresse redaction@elektor.fr.

Produits

- **OPEN-SMART GPS - Module GPS série pour Arduino (SKU 18733)**
  https://elektor.fr/18733
- **Module d'affichage TFT SPI 2,2 pouces ILI9341 (SKU 18419)**
  https://elektor.fr/18419
- **Majid Pakdel, *Advanced Programming with STM32 Microcontrollers* (SKU 19520)**
  https://elektor.fr/19520

LIENS

[1] Bibliothèque TinyGPS++ : http://arduiniana.org/libraries/tinygpsplus/
[2] Qu'est-ce que le POPD ? : https://fr.wikipedia.org/wiki/Geometric_dilution_of_precision
[3] Fichiers du projet sur Elektor Labs : https://elektormagazine.fr/labs/save-money-with-this-speed-monitoring-by-gps
[4] EDI Arduino : paquet de support des cartes STM32 : https://github.com/stm32duino/BoardManagerFiles/raw/main/package_stmicroelectronics_index.json