Projet Course en cours

Notice succincte pour Arduino V2020_01 (sans librairie fonctions_carte_mere_CEC.h)

Fichiers téléchargeables sur : https://uncloud.univ-nantes.fr/index.php/s/E9MTfJyc3BkJsQk

Préambule :

Ce document permet de rappeler à un utilisateur les liaisons de la carte moteur à travers le « connecteur élève » et propose quelques morceaux de programmes en C pour Arduino. Pas de fonction de lecture de capteur disponible à ce jour.

Au 20/01/2020 les librairies fonctions_carte_mere_CEC.h ne sont pas opérationnelles, la version « béta » à corriger est disponible pour les experts dans le dossier « fonctions_pour_arduino_non_fonctionnelles_a_deboguer ».

Vous trouverez dans ce document le schéma de câblage à respecter et les lignes de code pour faire démarrer la voiture avec les profils que vous rentrerez dans votre Arduino.

Les fichiers disponibles dans le répertoire sont :

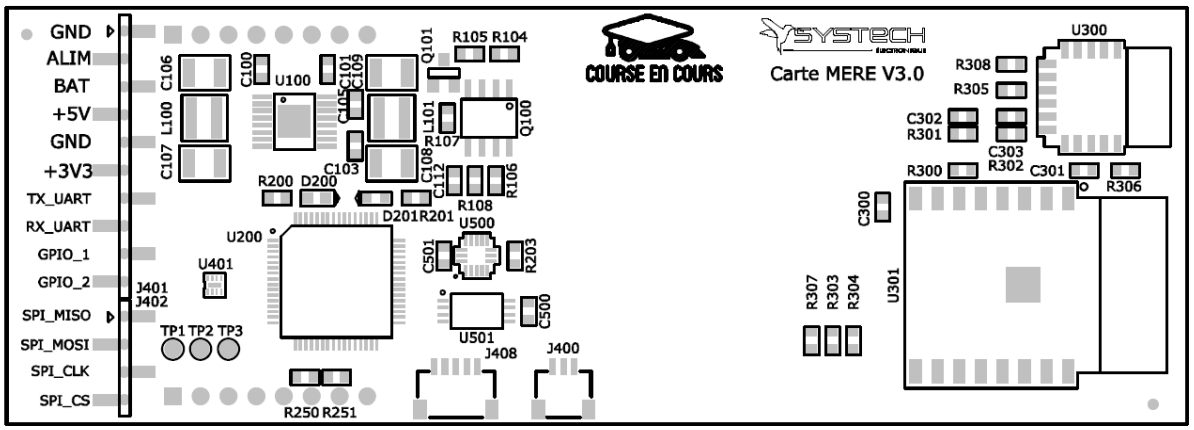
A lire impérativement avant branchement ARDUINO	Document à lire impérativement avant tout branchement
CeC - API Véhicule - V1.9	Fichier Excel avec les codes des trames
Code programmation carte moteur configuration action	Code source en C des exemples pour Arduino contenus dans ce document
fonctions_pour_arduino_non_fonctionnelles_a_deboguer	Dossier contenant une librairie pour Arduino à déboguer

Fichiers téléchargeables sur : https://uncloud.univ-nantes.fr/index.php/s/E9MTfJyc3BkJsQk

Attention : il faudra impérativement envoyer un signal en 3v3 à la carte mère sous peine de destruction.

Connecteur carte mère :

Les connecteurs J401 et J402 sont accessibles depuis l'extérieur du boitier.



Retrouvez le détail des signaux ci-dessous. Ne pas connecter les bornes ROUGES

- GND	**: Ground**
- ALIM	: ON/OFF de la carte mère commandé par la carte capteur
- BAT	**: Tension batterie +11,1V / 450mAh**
- +5V	**: Alimentation stabilisé +5V / 2A**
- GND	**: Ground**
- +3V3	**: Alimentation stabilisé +3,3V / 3A**
- Tx_UART 3v3	**: Communication 3V3 UART avec le système.**
- Rx_UART 3v3	**: Communication 3V3 UART avec le système**
- GPIO_1	: Non utilisé par le MCU
- GPIO_2	: Non utilisé par le MCU
- SPI_MISO	: Non utilisé par le MCU
- SPI_MOSI	: Non utilisé par le MCU
- SPI_CLK	: Non utilisé par le MCU
- SPI_CS	: Non utilisé par le MCU

Principe :

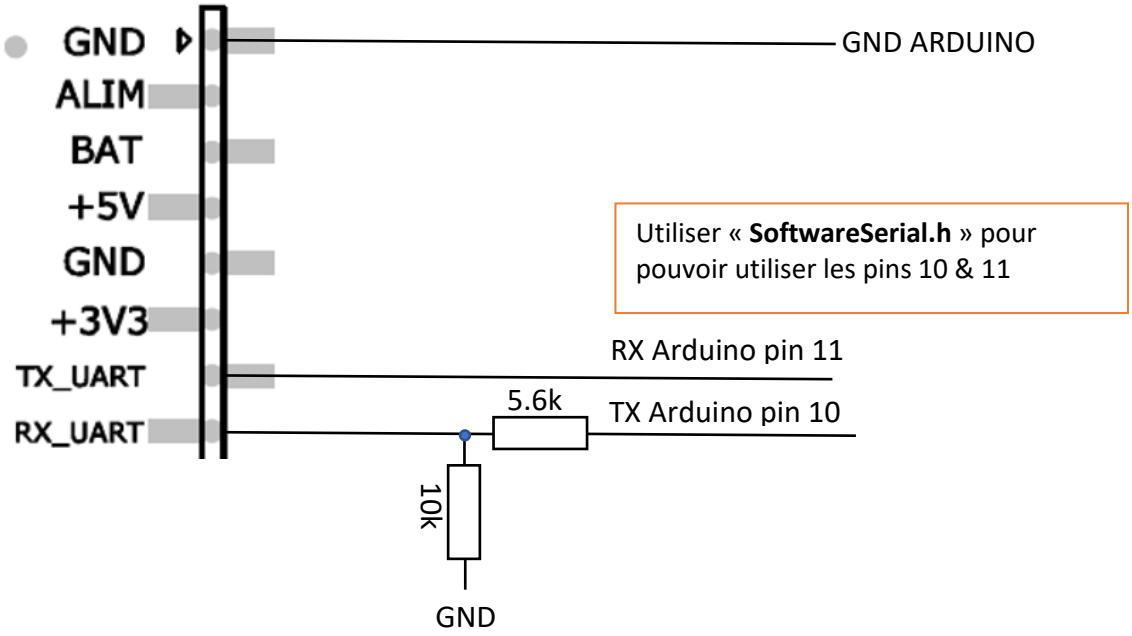
La carte moteur communique via une liaison série avec une carte Arduino Uno (ou autre système). Cette liaison s'établit sur deux broches RX et TX définies respectivement dans votre programme en tant que broche 11 et 10.

La broche RX de la carte Arduino Uno doit être raccordée à la broche TX de la carte moteur. De la même manière, la broche TX de la carte Arduino Uno doit être raccordée à la broche RX de la carte moteur (via un diviseur de tension).

Schéma électrique du montage :

Attention : il est nécessaire d'appliquer un pont diviseur de tension au niveau de la transmission de l'Arduino Uno (TX) vers la carte mère (RX). En effet, l'Arduino Uno envoie du 5V à partir de sa pin TX or la carte mère reçoit uniquement du 3,3V sur sa pin RX.

Connection entre un ARDUINO 5V (UNO…) et la carte moteur :



Partie logicielle :

Détails communication UART

- Vitesse : 115200 Baud
- Start : 1 bit
- Stop : 1 bit
- Octet : 8 bits

```
Code Arduino : mySerial.begin(115200); // Utiliser « SoftwareSerial.h »
```

Protocole de communication avec le système

Le document « CeC - API Véhicule – V1.9 » détaille le protocole à utiliser.
Pour communiquer avec le système à partir d'un Arduino, **nous conseillons d'utiliser l'ID 0x04** car il n'y a pas le CRC à calculer

Identifiant carte mére	
0x03	ID carte mère. CRC envoyé et contrôlé
0x04	ID carte mère. CRC envoyé et non contrôlé

Dans le logiciel Arduino :

- il faut ajouter la librairie série (soft) pour utiliser des pins différentes de la 0 et la 1 qui permettent le débogage :

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(11, 10); // (Pin RX VOITURE, Pin TX VOITURE)
```

- Il faut envoyer au moteur la trame de configuration ci-dessous qui est expliquée dans les onglets Définition protocole + Commandes + Trame – configurations du fichier « CeC - API Véhicule – V1.9 »

```
//configuration  :
  mySerial.write(0xA4); //pas de crc
  mySerial.write(0x02); //début de trame
  mySerial.write((byte)0x00); //nb octets de la configuration
  mySerial.write(69);
  mySerial.write(0x21); //commande configuration

  mySerial.write(0x07); //longueur piste = 20m (cm)
  mySerial.write(0xD0);

  mySerial.write(0x0A); //longueur damier = 10 (mm)
  mySerial.write(0x02); //diamètre roue = 5.5cm (dixième mm)
  mySerial.write(0x26);

  mySerial.write(0x0F); //nb dents pignon moteur
  mySerial.write(0x0F); //nb dents couronne axe roue
  mySerial.write((byte)0x00); //config vehicule
  mySerial.write((byte)0x00); //config course
  mySerial.write((byte)0x00); //réservé
  mySerial.write((byte)0x00); //périodicité des mesures
  mySerial.write((byte)0x00); //couleur ligne départ
  mySerial.write((byte)0x00); //couleur ligne intermédiaire 1
  mySerial.write((byte)0x00); //couleur ligne intermédiaire 2
  mySerial.write((byte)0x00); //couleur ligne arrivée

  mySerial.write(0x13); //durée course = 5s (ms) 5000=0x1388
  mySerial.write(0x88);

  mySerial.write((byte)0x00); //seuil vitesse min = 10cm/s (cm/s)
  mySerial.write(0x0A);

  mySerial.write(0x03); //seuil vitesse max = 10m/s (cm/s)
  mySerial.write(0xE8);

  mySerial.write(0x03); //vitesse vehicule zone 1 = 10m/s (cm/s)
  mySerial.write(0xE8);

  mySerial.write(0x0F); //temps zone 1 = 4s (ms)
  mySerial.write(0xA0);

  for (int i=0; i<=42; i++)
  {
    mySerial.write((byte)0x00);
  }

  mySerial.write((byte)0x00); //crc MSB
  mySerial.write((byte)0x00); //crc LSB
  mySerial.write(0x03); //fin de trame
```

Ensuite, pour faire tourner le moteur :

- Il faut envoyer au moteur la trame de demande de mouvement ci-dessous qui est expliquée dans les onglets Définition protocole + Commandes + Trame-actions du fichier « CeC - API Véhicule – V1.9 »

```
//action:
mySerial.write(0xA4); //pas de crc
mySerial.write(0x02); //début de trame
mySerial.write((byte)0x00); //nb octets de l'action
mySerial.write(13);
mySerial.write(0x30); //commande action

mySerial.write(0x02); //actions véhicule
mySerial.write((byte)0x00); //heure start
mySerial.write((byte)0x00); //minute start
mySerial.write((byte)0x00); //seconde start
mySerial.write((byte)0x00); //milliseconde start
mySerial.write((byte)0x00);

mySerial.write((byte)0x00); //réservé
mySerial.write((byte)0x00);

mySerial.write((byte)0x00); //réservé
mySerial.write((byte)0x00);

mySerial.write((byte)0x00); //réservé
mySerial.write((byte)0x00);

mySerial.write((byte)0x00); //crc MSB
mySerial.write((byte)0x00); //crc LSB
mySerial.write(0x03); //fin de trame
while(1);//bloquage du programme
```