

Commande et instrumentation de trottinette électrique 500W avec Arduino méga

Trottinette Electrique 500W avec Arduino méga

 Difficulté Moyen

 Durée 1 jour(s)

 Catégories Électronique, Énergie, Transport

 Coût 1 EUR (€)

Sommaire

Introduction

Étape 1 - 2. Bibliographie :

Étape 2 - caractéristique et programme en boucle ouverte précédent

Étape 3 - 10. mesure de la vitesse (tachymetre)

Étape 4 - en cours d'écriture :

Commentaires

Introduction

Commande de moteur DC 500W avec un Arduino mega pour limiter le courant de démarrage et faire varier la vitesse de la trottinette. La batterie est en 24V, 10A.h. le tableau suivant résume leur caractéristiques

3. Programme en boucle ouverte

Pour tester la programmation, nous simulons le programme dans ISIS, comme on peut le voir sur la figure suivante. De plus, nous avons un afficheur LCD pour afficher des données (rapport cyclique correspondant à la PWM à 32Khz, le courant moteur, la tension moteur, l'action sur les boutons poussoirs. En effet, 4 boutons poussoirs sont utilisés.

BP1 pour incrémenter manuellement le rapport cyclique, BP2 le décrémenter. BP3 mettre le rapport cyclique à 0, correspondant au contact frein.

La vitesse du moteur est pratiquement proportionnelle au rapport cyclique

<https://i58.servimg.com/u/f58/17/56/35/17/a211.jpg>

Nous avons réalisé notre propre amplificateur de courant qui s'appelle un hacheur abaisseur mais il est possible d'acheter un shield

Il existe de nombreuses cartes pour Arduino pour commander des moteurs DC surtout de faibles puissances et aussi de grandes puissances comme on peut l'observer sur les liens suivants.

http://www.robotpower.com/products/MegaMotoPlus_info.html

<http://www.robotshop.com/en/dc-motor-driver-2-15a.html>

<https://www.pololu.com/file/0J51/vnh3sp30.pdf>

<https://i58.servimg.com/u/f58/17/56/35/17/a310.jpg>

mais, tous ces hacheurs shields mesurent le courant en interne mais il n'y a pas de limitation de courant.

Pour avoir une limitation de courant il faut une boucle de courant analogique en utilisant des AOP ou CI spécialisée ou une boucle de courant numérique rapide.

Mais quel doit être la valeur du courant de limitation ?

Le choix de la valeur du courant est normalement pour le Service de fonctionnement 1 heure pour pouvoir effectuée des montées relativement longue sans atteindre la température critique du moteur.

Dans notre cas, le courant de limitation devra être de

$I_{\text{moteur limitation}} = \text{Puissance} / U_{\text{batterie}} = 500W / 24V = 20A$

De plus, le transistor de puissance du hacheur ne peut supporter que 50A dans notre cas.

Mais en boucle ouverte, il n'a pas de régulation de courant, pour ne pas avoir de dépassement du courant maximum, une rampe du rapport cyclique sera utilisé.

Une routine d'interruption de 0.1 seconde sera utilisé pour faire la mesure de la tension est du courant (échantillon de mesure, sample). Ce temps de sampler est arbitraire, mais ne permet pas d'être plus rapide que le temps de montée du courant car la constante de temps électrique du moteur étant de $L/R = 1.5ms$.

Le fonctionnement en boucle ouverte avec une rampe de 25.5s (8bit et routine d'interruption de 0.1s) permet de bien comprendre la problématique du fonctionnement d'une commande à moteur DC.

l'affichage se fera seulement tous les 0.2s pour avoir une stabilité des chiffres à l'écran. De plus, un filtrage numérique, se fera sur le courant et la tension sur 4 valeurs donc sur 0.4s.

Algo boucle ouverte

Routine d'interruption toutes les 0.1s

Lire tension et courant

Boucle loop (scrutation des boutons poussoirs)

Si BP1=1 alors incrementer PWM

Si BP2=1 alors decrementser PWM

Si BP3=1 alors PWM=0

Affichage des variables tous les 0.2s

code

```
{  
// include the library code:  
#include <LiquidCrystal.h>  
#include <SoftwareSerial.h>  
#include <TimerOne.h>  
#define SERIAL_PORT_LOG_ENABLE 1  
#define Led 13 // 13 pour la led jaune sur la carte  
#define BP1 30 // 30 BP1  
#define BP2 31 // 31 BP2  
#define BP3 32 // 32 BP3  
#define LEDV 33 // 33 led  
#define LEDJ 34 // 34 led  
#define LEDR 35 // 35 led  
#define relay 36 // 36 relay  
#define PWM10 10 //11 timer2  
LiquidCrystal lcd(27, 28, 25, 24, 23, 22); // RS=12, Enable=11, D4=5, D5=4, D6= 3, D7=2, BPpoussoir=26  
// Configuration des variables  
unsigned int UmoteurF = 0; // variable to store the value coming from the sensor  
unsigned int Umoteur = 0;  
unsigned int Umoteur2 = 0;  
unsigned int Umoteur3 = 0;  
unsigned int Umoteur4 = 0;  
unsigned int ImoteurF = 0;  
unsigned int Imoteur = 0;  
unsigned int Imoteur2 = 0;  
unsigned int Imoteur3 = 0;  
unsigned int Imoteur4 = 0;  
byte Rcy=0; //rapport cyclique 8bit  
unsigned int temps;  
// the setup function runs once when you press reset or power the board  
void setup() {  
pinMode(Led, OUTPUT); //led carte arduino  
pinMode(LEDV, OUTPUT);  
pinMode(LEDR, OUTPUT);  
pinMode(LEDJ, OUTPUT);  
pinMode(PWM10, OUTPUT); // broche (10) en sortie timer2  
// digitalWrite(LEDV, LOW);  
Timer1.initialize(100000); // initialize timer1, and set a 0,1 second period => 100 000  
Timer1.attachInterrupt(callback); // attaches callback() as a timer overflow interrupt  
lcd.begin(20, 4);  
Serial1.begin(9600);  
TCCR2B = (TCCR2B & 0b11111000)
```

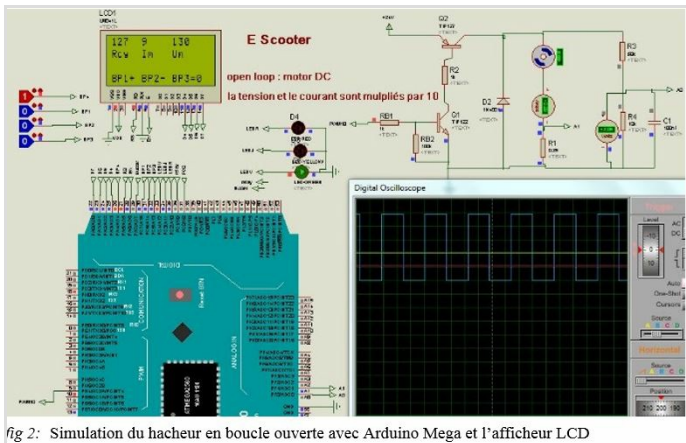


fig 2: Simulation du hacheur en boucle ouverte avec Arduino Mega et l'afficheur LCD

Matériaux

Outils

Étape 1 - 2. Bibliographie :

2. Bibliographie :

Lien download :

sketch_escooter_feed_back_reel_V1.ino

https://drive.google.com/file/d/0B_fB3GAsM02FSIRTWHdyRkhuUW8/view?usp=sharing

escooter_ampli_SIMULINK.mdl

https://drive.google.com/file/d/0B_fB3GAsM02FOW9OdmlhdDhJZGc/view?usp=sharing

escooter feed back ISIS.DSN

https://drive.google.com/file/d/0B_fB3GAsM02FOXdRWFN5OWRMQkE/view?usp=sharing

En anglais

<https://forum.arduino.cc/index.php?topic=477397.0>

article : « Etude de trottinettes électriques 100W et 500W (Arduino), Revue 3EI 2017 »

En attente

3. Programme en boucle ouverte

Pour tester la programmation, nous simulons le programme dans ISIS, comme on peut le voir sur la figure suivante. De plus, nous avons un afficheur LCD pour afficher des données (rapport cyclique correspondant à la PWM à 32Khz, le courant moteur, la tension moteur, l'action sur les boutons poussoirs. En effet, 4 boutons poussoirs sont utilisés.

BP1 pour incrémenter manuellement le rapport cyclique, BP2 le décrémenter. BP3 mettre le rapport cyclique à 0, correspondant au contact frein.

La vitesse du moteur est pratiquement proportionnelle au rapport cyclique

<https://i58.servimg.com/u/f58/17/56/35/17/a211.jpg>

Nous avons réalisé notre propre amplificateur de courant qui s'appelle un hacheur abaisseur mais il est possible d'acheter un shield

Il existe de nombreuses cartes pour Arduino pour commander des moteurs DC surtout de faibles puissances et aussi de grandes puissances comme on peut l'observer sur les liens suivants.

http://www.robotpower.com/products/MegaMotoPlus_info.html

<http://www.robotshop.com/en/dc-motor-driver-2-15a.html>

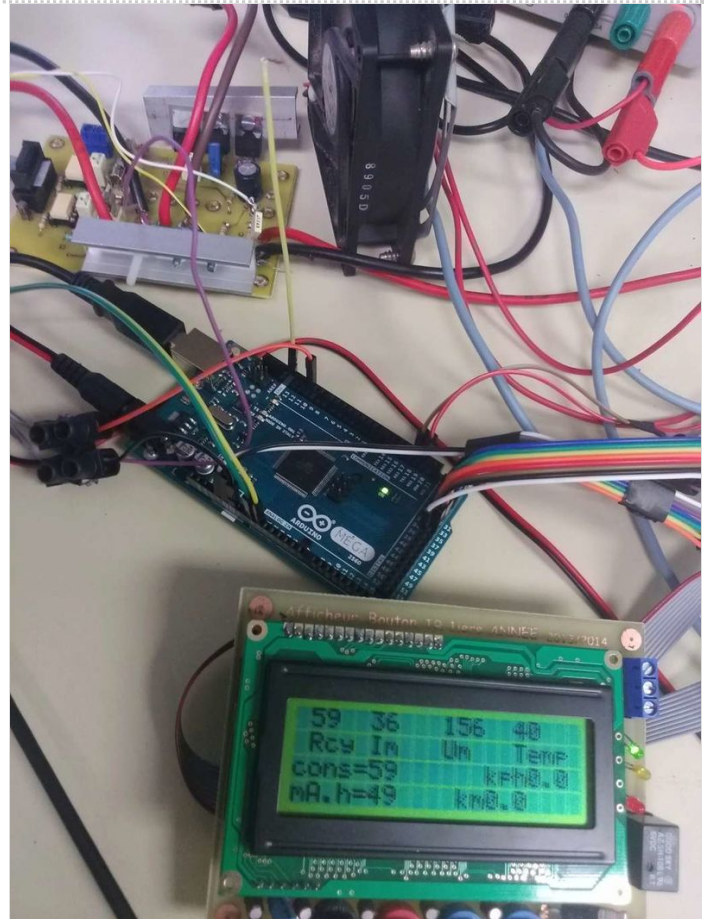
<https://www.pololu.com/file/0J51/vnh3sp30.pdf>

<https://i58.servimg.com/u/f58/17/56/35/17/a310.jpg>

mais, tous ces hacheurs shields mesurent le courant en interne mais il n'y a pas de limitation de courant.

Pour avoir une limitation de courant il faut une boucle de courant analogique en utilisant des AOP ou CI spécialisée ou une boucle de courant numérique rapide.

Mais quel doit être la valeur du courant de limitation ?



Le choix de la valeur du courant est normalement pour le Service de fonctionnement 1 heure pour pouvoir effectuée des montées relativement longue sans atteindre la température critique du moteur.

Dans notre cas, le courant de limitation devra être de

$I_{\text{moteur limitation}} = \text{Puissance} / U_{\text{batterie}} = 500\text{W} / 24\text{V} = 20\text{A}$

De plus, le transistor de puissance du hacheur ne peut supporter que 50A dans notre cas.

Mais en boucle ouverte, il n'a pas de régulation de courant, pour ne pas avoir de dépassement du courant maximum, une rampe du rapport cyclique sera utilisé.

Une routine d'interruption de 0.1 seconde sera utilisé pour faire la mesure de la tension et du courant (échantillon de mesure, sample). Ce temps de sampler est arbitraire, mais ne permet pas d'être plus rapide que le temps de montée du courant car la constante de temps électrique du moteur étant de $L/R = 1.5\text{ms}$.

Le fonctionnement en boucle ouverte avec une rampe de 25.5s (8bit et routine d'interruption de 0.1s) permet de bien comprendre la problématique du fonctionnement d'une commande à moteur DC. L'affichage se fera seulement tous les 0.2s pour avoir une stabilité des chiffres à l'écran. De plus, un filtrage numérique, se fera sur le courant et la tension sur 4 valeurs donc sur 0.4s.

Algo boucle ouverte

Routine d'interruption toutes les 0.1s

Lire tension et courant

Boucle loop (scrutation des boutons poussoirs)

Si BP1=1 alors incrémenter PWM

Si BP2=1 alors décrémenter PWM

Si BP3=1 alors PWM=0

Affichage des variables tous les 0.2s

code

```
{{
```

```
// include the library code:
```

```
#include <LiquidCrystal.h>
```

```
#include <SoftwareSerial.h>
```

```
#include <TimerOne.h>
```

```
#define SERIAL_PORT_LOG_ENABLE 1
```

```
#define Led 13 // 13 pour la led jaune sur la carte
```

```
#define BP1 30 // 30 BP1
```

```
#define BP2 31 // 31 BP2
```

```
#define BP3 32 // 32 BP3
```

```
#define LEDV 33 // 33 led
```

```
#define LEDJ 34 // 34 led
```

```
#define LEDR 35 // 35 led
```

```
#define relay 36 // 36 relay
```

```
#define PWM10 10 //11 timer2
```

```
LiquidCrystal lcd(27, 28, 25, 24, 23, 22); // RS=12, Enable=11, D4=5,
```

```
D5=4, D6= 3, D7=2, BPpoussoir=26
```

```
// Configuration des variables
```

```
unsigned int UmoteurF = 0; // variable to store the value coming  
from the sensor
```

```
unsigned int Umoteur = 0;
```

```
unsigned int Umoteur2 = 0;
```

```
unsigned int Umoteur3 = 0;
```

```
unsigned int Umoteur4 = 0;
```

```
unsigned int ImoteurF = 0;
```

```
unsigned int Imoteur = 0;
```

```
unsigned int Imoteur2 = 0;
```

```
unsigned int Imoteur3 = 0;
```

```
unsigned int Imoteur4 = 0;
```

```
byte Rcy=0; //rapport cyclique 8bit
```

```
unsigned int temps;
```

Étape 2 - caractéristique et programme en boucle ouverte précédent

```

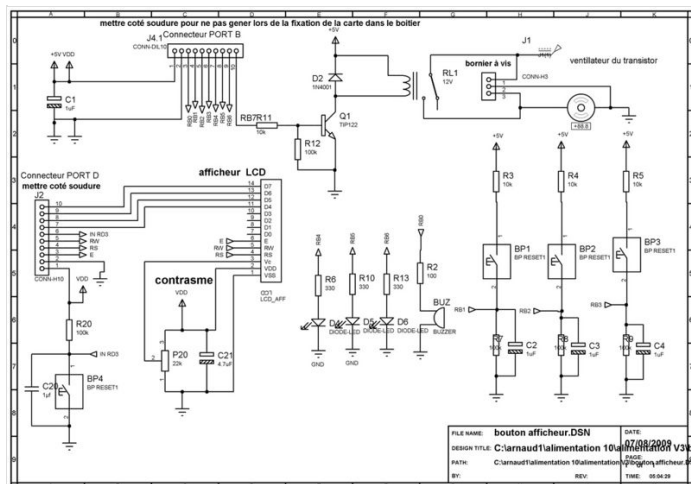
if (Imoteur<4000) // pas de limitation de courant à (20A*10)*20=4000
{if (consigne>Rcy) {Rcy=Rcy+1;} // rampe de de la Pwm +1*0.01seconde integrateur pur
if (consigne<Rcy && Rcy!=0) {Rcy=Rcy-1;} //la decrementation est faite seulement pour la poignée d'acceleration ou avec BP2
if ( Rcy>254) {Rcy=254;} //limitation du rapport cyclique
analogWrite(PWM10,Rcy); //frequence 32kHz timer2}
}

if (Imoteur>4000) { Rcy=Rcy-5; //pas de filtrage du courant, pour etre plus rapide
if ( Rcy<6) {Rcy=5;} //rcy n'est pas signé, ni la PWM donc Rcy minimun ne doit pas etre inferieur à 6
analogWrite(PWM10,Rcy); //frequence 32kHz timer2}

```

Étape 3 - 10. mesure de la vitesse (tachymetre)

<https://i58.servimg.com/u/f58/17/56/35/17/a018.jpg>



code

```
void INT0b21() {
Tspeed++; //interruption exterieure pour compter le nombre de
tour
}
//dans le set up declarer la routine d'interruption lorsque le front 5V
de la detection de l'aimant se fait
attachInterrupt(digitalPinToInterrupt(21), INT0b21, RISING );
//interruption exterieur
//dans loop
if (temps09>=5) { //boucle de 1 seconde
lcd.setCursor(13,2); // effacement de la vitesse
lcd.print("kph ");
lcd.setCursor(16,2);
speed1=Tspeed*2937; //1tour*816*3.6/1s=2.937km/h
speed2=speed1; //Tspeed (rate/seconde)
speed3=speed2;
speedF=(speed1+speed2+speed3)/3000; //pour mettre en kph
lcd.print(speedF,1); //affichage au dixieme pres
Tspeed=0; //reset compteur
temps09=0; //reset time
}
```

Pour améliorer la précision de la mesure de la vitesse, il est possible que le temps échantillonnage de la mesure de la vitesse soit en fonction de la vitesse.

exemple :

pour les vitesses inferieures à 10km/h echantillon à 1seconde, mais au dessus de 10km/h echantillon à 2 secondes.

11. Mesure distance pour connaitre l'autonomie

La distance correspond au nombre de tour total de la roue multipliée par le périmètre de la roue.

Donc il ne faut pas remettre à 0, le nombre de tour à chaque échantillon.

Par contre, la remise à zéro de la distance sera effectuée lors l'appuie sur le reset de l'Arduino Mega.

L'affichage de la distance s'effectuera au deuxième près.

A 32km/h, il faudra 2 minutes pour faire 1km comme on peut l'observer sur la figure suivante :

<https://i58.servimg.com/u/f58/17/56/35/17/a019.jpg>

code

```
void INT0b21() {
Tspeed++; //interruption exterieure pour compter la vitesse
nbrRate++;
}
lcd.setCursor(13,4);
lcd.print("km "); //
distance=(nbrRate*816)/1000; //distance m
distance=distance/1000; //distance km
lcd.setCursor(15,4);
lcd.print(distance,1);
on peut observer l'installation Electrique avec le hacheur, l'arduino,
et l'afficheur lors de la mise au point du programme
https://i58.servimg.com/u/f58/17/56/35/17/dsc\_0613.jpg
```

12. Synthèse

L'espace RAM est utilisé que à 4% et l'espace ROM à 3%, pour un Arduino mega. Donc, on pourrait prendre un arduino un peu plus petit.

Mais, il y a 8 cellules Lipo pour faire l'alimentation 24V pour alimenter le moteur via le hacheur. Par conséquent, la mesure de la tension de chaque élément sera sur l'Arduino avec un connecteur JST. Cette mesure permet de savoir si une cellule à une résistance interne qui commence à poser problème et pour savoir si l'équilibrage de chaque cellule a bien été effectué.

Il est possible de passer à 36V avec 12 cellules aussi avec l'arduino mega sans utiliser de shield extérieur qui multiplexe 24 entrées analogiques sur l'entrée A0

Il est possible d'envoyer toutes les données à un smartphone via le bluetooth HC06 par les broches 20, 21, RX1 et TX1. Mais

L'application sous android réalisée sous JAVA Studio ne peut pas être partagée sur ce forum. Cette partie ne sera pas explicitée.

Après avoir fait l'instrumentation de cette trotinette, une étude devrait être effectuée sur la précision des mesures, il est possible de lire

« Instrumentation d'un véhicule motorisé électrique faible consommation de type « éco marathon » Revue 3EI N°81, Juillet 2015

<http://www.fichier-pdf.fr/2015/09/07/instrumentation-vehicule-faible-consommation-eco-marathon/>

Étape 4 - en cours d'écriture :

Voici les schémas électriques, 3 cartes sont utilisées

une carte qui va s'afficher dans l'arduino, mesure température, courant, et qui fait le lien vers la carte afficheur

https://i58.servimg.com/u/f58/17/56/35/17/pont_d11.jpg

voici le typon

https://i58.servimg.com/u/f58/17/56/35/17/pont_d10.jpg

une carte 4 boutons poussoirs et un afficheur LCD recouper d'ancien système microcontrôleur qui utilisait des connecteurs HE10

<https://i58.servimg.com/u/f58/17/56/35/17/bouton10.jpg>

voici le typon double face

<https://i58.servimg.com/u/f58/17/56/35/17/bouton11.jpg>

une carte hacheur qui va s'enficher dans la première carte.

le schéma électrique de la commande du hacheur n'a rien à voir avec la carte réelle car l'optocoupleur n'est pas simulable, et il n'y a pas de modèle spice de nos composants

à la place d'utiliser, un optocoupleur un driver IRF2110 est préférable d'utilisation.

le condensateur entre le Drain et la source permet de minimiser les oscillations lors des commutations.

de plus, il permet de minimiser les pertes à la commutations dans le transistor.

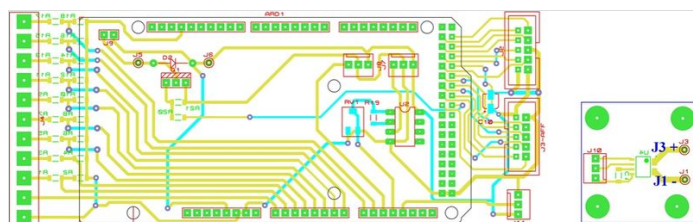
<https://i58.servimg.com/u/f58/17/56/35/17/hacheu10.jpg>

vidéo

youtube : "study trotinette electric e-scooter 100W et 350W, wiring" https://www.youtube.com/watch?v=QqJ2-YiE8Tg&list=PLfZunVn_gcq7EOurXuWU2sRFmh6CbiUiL&index=75

<https://www.google.fr/search?q=sepduino&oq=sepduino&aqs=chrome..69i57j0l5.15544j0j7&sourceid=chrome&ie=UTF-8#q=sepduino>

un livre « je réalise mon véhicule électrique » chez DUNOD



5. Programme en boucle fermée, commande à courant limité avec poignée d'accélération

Une poignée d'accélération fournit une tension 0.8V lorsqu'elle n'est pas actionnée et une tension 4.5V lorsque la poignée est à fond.

À la place d'utiliser des boutons poussoirs pour augmenter ou diminuer la consigne de vitesse, une poignée d'accélération sera donc utilisée

Upoignee=analogRead(A3); //la relation en Upoignée et la consigne qui correspondra au rapport cyclique correspond à

code

```
if (Upoignee > 100) { consigne = (Upoignee / 2); //0=a*200+b et 255=a*800+b
consigne = consigne - 100;
}
else { consigne = 0; }
```

```
if (Upoignee < 100) { consigne = 0; } //redondance
```

6. Programme mesure température et sécurité du moteur avec la mesure courant

La mesure de température extérieure peut être facilement effectuée par le composant LM35 qui fournit 0.01V par degrés Celsius

code

```
temperature=analogRead(A2); //lm35 0.01V/°C
temperature=temperature/2; //coefficient de mesure de temperature
lcd.setCursor(5,2);
lcd.print(" ");
lcd.setCursor(5,2);
lcd.print(temperature); // affichage en °C
lcd.setCursor(9,2); // effacement de l'affichage secu
lcd.print(" ");
if (temperature>80 ){lcd.setCursor(9,2); // si temperature extern moteur est superieur à 80°C
lcd.print("secuT");
Rcy=0;}
```

De plus, une sécurité thermique par la mesure du courant moteur sera ajoutée.

si le courant de limitation est superieur à 10s alors le moteur ne sera plus alimenté pendant 30s.

un affichage "secu" sur l'afficheur LCD sera indiqué.

Cette sécurité permet de couper le moteur lors de pente trop importante et lors du blocage du moteur mais il faudrait rajouter la mesure de la vitesse dans ce dernier cas

code

```
if (timesecurite>=10000 ){flagarret=1;
// si courant de limitation pour un courant de plus de 10s
timerepos=0;
consigne=0;
Rcy=0;
timesecurite=0;} // alors arret moteur pendant un temps d'arret
if (flagarret==1 ){lcd.setCursor(9,2); // si courant de limitation pour un courant de plus de 20s
lcd.print("secU"); } // alors arret moteur pendant 60 un temps d'arret et affichage
if (timerepos>=30000 && flagarret==1){flagarret=0;
lcd.setCursor(9,2); // apres un temps de repos ici de 30s
lcd.print(" "); }
```

on peut observer l'affichage si la température est supérieure à 80°C

<https://i58.servimg.com/u/f58/17/56/35/17/a017.jpg>

Une sécurité thermique par la mesure du courant moteur (relais thermique numérique) qui permet de connaître l'image de la température interne du moteur serait idéale. Mais pour cela, il faut bien connaître le modèle thermique du moteur.

7. Programme mesure de la capacité énergétique de la batterie

La capacité énergétique d'une batterie est en A.H, nous afficherons la valeur en mA.H pour avoir une grande précision. La capacité sera en A.Seconde dans l'équation suivante. Donc pour avoir en mA.H, il faudra diviser par la capacité par 3600.

Capacité(A.s)_n=I*Te+C_{n-1} avec Te=0.01s et I multiplié par 10

Donc dans la routine d'interruption

code

```
capacity=ImoteurF+capacity ;
et dans l'affichage
lcd.setCursor(0,3); //affichage de la capacité énergétique
lcd.print("C mA.h=");
capacity1=capacity/(18000); //18000=3600*5 5=>coefficient mesure courant
lcd.print(capacity1);
```

pour vérifier mettre un courant de 10A avec une résistance ajustable et au bout de 30s, la capacité devra être de 83mA.H

8. Bilan puissance et modélisation avec SIMULINK

La modélisation permet de bien comprendre le véhicule et sa commande. De plus, il est possible de compiler la partie régulation directement en programme Arduino à partir de la simulation sous Simulink. Mais il ne sera pas possible de simuler l'instrumentation avec l'afficheur LCD.

Sur la figure suivante, on peut observer la simulation de la programmation du hacheur avec la limitation de courant avec Simulink. Sur la figure suivante, l'encadrée en vert montre la commande du rapport cyclique pour faire varier la vitesse et l'encadrée en rouge la limitation du courant. Le correcteur de la régulation est ici un simple intégrateur mais il est possible d'effectuer une multitude de commande.

https://i58.servimg.com/u/f58/17/56/35/17/azub_c15.jpg

Sur la figure précédente, on peut observer que le courant est bien limité à 25A de 2s à 9.5s. Puis, le courant atteint 10.8A en régime établi de vitesse à 22.5km/h. les dynamiques sont similaires aux essais effectués.

Avec une pente de 5%, le rapport cyclique n'atteint tout juste 100% comme on peut l'observer sur la figure suivante. La vitesse atteindra péniblement 19km/h avec un courant de 24A et une puissance moteur de 580W. Voir l'article : Etude de trottinettes électriques 100W et 500W (Arduino), **9. Première conclusion** Il est facile de commander un moteur DC de 500W avec un Arduino et quelques composants. Donc de réparer nombreuses trottinettes qui sont en moteur DC. mais il faut quelques connaissances (automatique, moteur) pour savoir gérer correctement le moteur et limiter son courant pour ne pas l'endommager. L'affichage de la vitesse, la distance, de l'heure de fonctionnement pour connaître les Watt.heure/km pourra être aussi réalisé avec un menu 2. le programme .ino en fichier attaché, mais il n'est pas possible de

mettre un fichier attaché .dns de ISIS labcenter electronic ? c'est quoi ce forum ! il serait souhaitable que le compilateur puisse générer le.cof pour pouvoir debugger dans isis et tester le programme ligne par ligne.... Arduino doit encore faire beaucoup d'effort pour être au même niveau que d'autre microcontrôleur

<https://forum.arduino.cc/Themes/default/images/icons/clip.png>

8.47 KB

downloaded 10 times

|Step_Picture_00=Commande_et_instrumentation_de_trottinette_électrique_500W_avec_Arduino_méga_st4.jpg}}