GEII
Département Génie Électrique
& Informatique Industrielle
IUT Belfort-Montbéliard

HOME-TRAINER

RAPPORT DE PROJET TUTORÉ

Année 2017-2018

Robin FERNANDEZ

Lucas MARINHO

Licence Professionnelle VEGA

Véhicules : Électronique &
Gestion des Automatismes

HILAIRET Mickaël

UFC
UNIVERSITÉ
DE FRANCHE-COMTÉ

IUT
Belfort-
Montbéliard

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE DE BELFORT-MONTBÉLIARD
Tél. +33 (0)3 84 58 77 00 • iut-bm.univ-fcomte.fr

Remerciements

Nos remerciements vont à notre encadrant technique Mr HILAIRET pour son aide au lancement du projet et qui nous a aidé durant le déroulement de celui-ci. Ses conseils nous ont permis de nous orienter dans la bonne direction, ce qui a était extrêmement instructif.

Nous remercions également toutes l'équipe pédagogique du département GEII de l'IUT Belfort-Montbéliard et les intervenants professionnels responsables de la formation licence professionnelle VEGA : Véhicules Électroniques et Gestion des Automatismes pour avoir assuré la partie théorique de celle-ci, Mr Frédéric GUSTIN, enseignant-chercheur, pour sa contribution, ainsi que Mr Vincent HUBERT pour la maintenance du Home trainer.

Table des matières

Introduction

Notre produit, home trainer, est un appareil permettant aux personnes d'utiliser leur vélo pour s'entraîner chez elles lorsque les conditions climatiques ne permettent pas de le faire à l'extérieur. En fait, l'entraînement sur home trainer est plus efficace que sur route. Il n'y a pas de descentes ni de moments de relâche. Une demi-heure de travail correspond approximativement à 1 heure de vélo sur route.

Peu encombrant, simple d'utilisation, ne demandant aucune préparation spécifique, le Home trainer facilite par ailleurs les séances en hiver quand la tombée de la nuit ou la température extérieure rendant une sortie difficile. Les cyclistes professionnels ou amateurs peuvent s'équiper de cet appareil.

Il existe différents types de technologie :

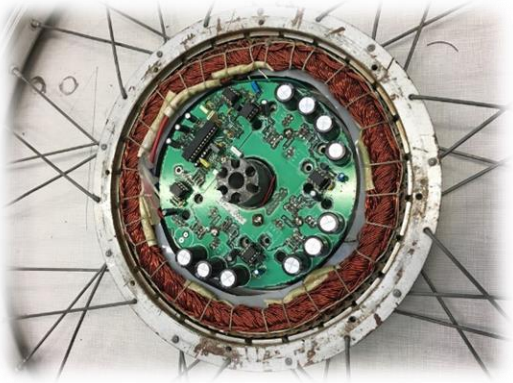
Le home-trainer à transmission directe : il permet de relier directement la chaine du vélo à la machine équipée d'une cassette et non à la roue arrière complète. Il peut simuler des pentes jusqu'à 18% et ce, de manière discrète.

Le home-trainer à rouleaux : il permet de travailler l'équilibre, la technique de pédalage et la vélocité. Le vélo est posé sur des rouleaux mais n'est pas fixé. Il ne permet donc pas de simuler des pentes ni de travailler la force et la puissance.

Le home-trainer à résistance : il permet de fixer la roue arrière du vélo directement sur le home trainer, on peut régler le niveau de résistance en ajustant le frein électromagnétique.

Présentation du projet :

Nous sommes partis d'un vélo électrique en utilisant le moteur positionné sur la roue arrière en alternateur ainsi nous avons pu gagner en fluidité de pédalage et il ne subira pas d'usure mécanique dans le temps grâce au moteur brushless (sans balais). La charge alimentée par l'alternateur grâce à sa résistance impacte le couple développé.



Le vélo est positionné sur un châssis qui a été amélioré pour pédaler dans de meilleures conditions.

L'utilisateur pourra observer la puissance instantanée grâce à un afficheur tactile ainsi que d'autres fonctionnalités.

Le projet se déroule en plusieurs tâches :

- Récupérer la puissance développée, et convertir la donnée pour pouvoir la visualiser et la contrôler sur un écran tactile de marque Nextion.
- Intégrer un programme Arduino dans la carte DE1_SoC pour communiquer les informations vers l'écran tactile.
- Développer une interface IHM.
- Afficher la fréquence de pédalage.

Gestion de projet

Nos diverses expériences professionnelles et personnelles au cours de ces années d'études, nous ont permis d'aiguiser notre curiosité et de nous ouvrir à d'autres domaines et technologies. Nos centres d'intérêts, souvent abordés lors de nos discussions quotidiennes, nous ont finalement rassemblés tous les deux cette année autour de ce projet, principalement axés sur la programmation afin de continuer les travaux des années précédentes.

La planification du projet et les outils de gestion

Cahier des charges

Le cahier des charge (ou objectif) fournit par notre professeur référent au début de l'année, décrit les conditions attachées à l'exécution du projet, cela nous a permis dans un premier temps de définir le contexte, les enjeux, les objectifs techniques ainsi que de nous pousser à comprendre le travail qui avait été fait auparavant. En organisant nos idées, nous avons ainsi pu vérifier la concordance et la faisabilité des différentes tâches.

Diagramme GANTT

Afin de gérer au mieux le temps imparti pour la réalisation de notre projet, nous avons réalisé un diagramme de GANTT en avant-projet, et avons tenté de nous y tenir depuis.

Objectifs et contraintes :

a. Les objectifs techniques

Étant pour le moment dans un projet universitaire limité en moyens et en temps, nous devons améliorer le projet réalisé par des étudiants de l'année précédente en sélectionnant des solutions à développer parmi toutes les possibilités. Notre objectif est de remplacer l'écran déjà existant, piloté par un FPGA installé sur le Home trainer, par un écran tactile de la marque Nextion piloté par un Arduino. Il servira d'interface homme-machine, et permettra de piloter les différents niveaux de pédalage ainsi que d'afficher différentes informations telles qu'un compteur de puissance.

b. Les objectifs économiques

Comme évoqué précédemment, un des enjeux du projet est économique. Un prix « maitrisé » est synonyme de produit grand public. L'écran Nextion est contrôlable par le biais d'une liaison série RS232 compatible avec l'Arduino, déjà existant ce qui implique de ne pas ajouter un FPGA qui rendrait le système plus coûteux.

c. Le délai

Afin de terminer ce projet ambitieux dans les temps, il est important de le gérer correctement et de le tenir à jour grâce aux outils de gestion adéquats. Dans cette optique, nous avons utilisé des outils autant présents dans le domaine universitaire que dans le monde du travail : journal de bord, diagramme de Gantt, diagramme Pert, etc.

d. Contraintes matérielles

Nous continuons un travail déjà entamé, nous devons donc utiliser l'existant, ce qui implique des contraintes matérielles, et logiciels, comme par exemple le logiciel Arduino :

Arduino est un circuit imprimé en matériel libre sur lequel se trouve un microcontrôleur qui peut être programmé pour analyser et produire des signaux électriques, de manière à exécuter des tâches très diverses comme le pilotage d'un robot. C'est une plate-forme basée sur une interface entrée/sortie simple. Le logiciel de programmation des modules Arduino est une application Java multi-plateforme servant d'éditeur de code et de compilateur. Celui-ci permet de transférer le programme au travers de la liaison série. Le langage de programmation utilisé est le C++, lié à la bibliothèque de développement Arduino, permettant l'utilisation de la carte et de ses entrées/sorties.

Diagramme bête à cornes :

Nous traçons le diagramme bête à cornes ci-dessous afin d'expliciter l'exigence fondamentale qui justifie la conception de notre produit.

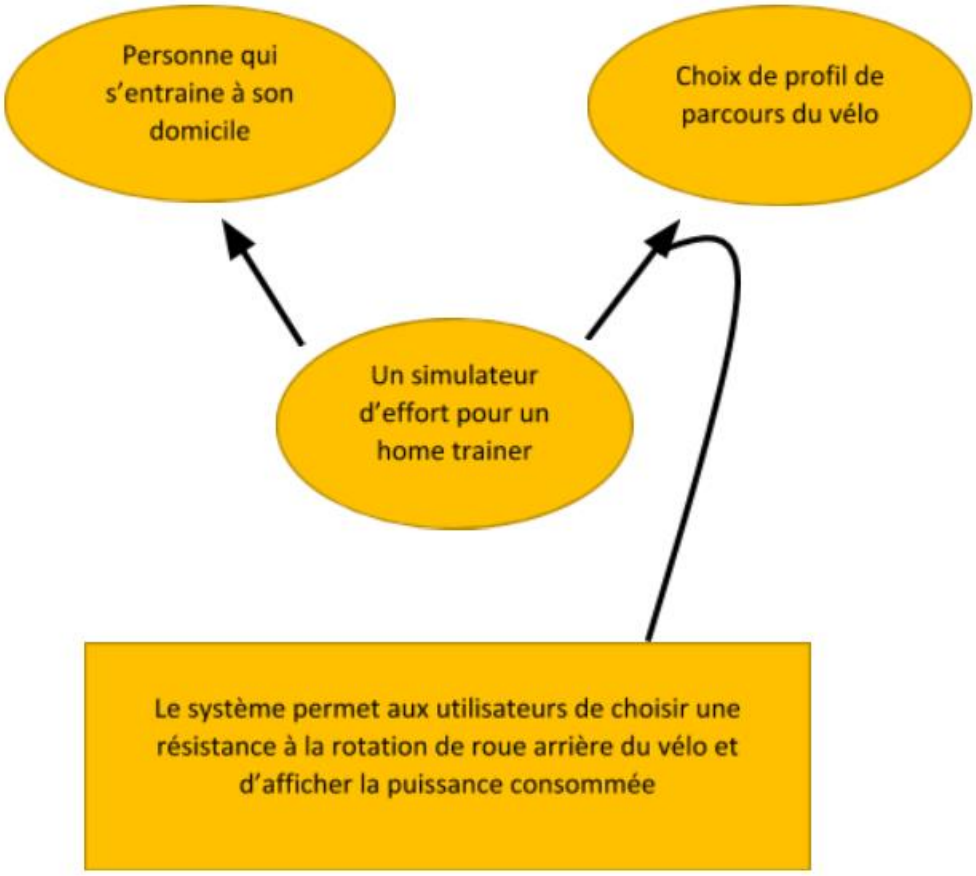
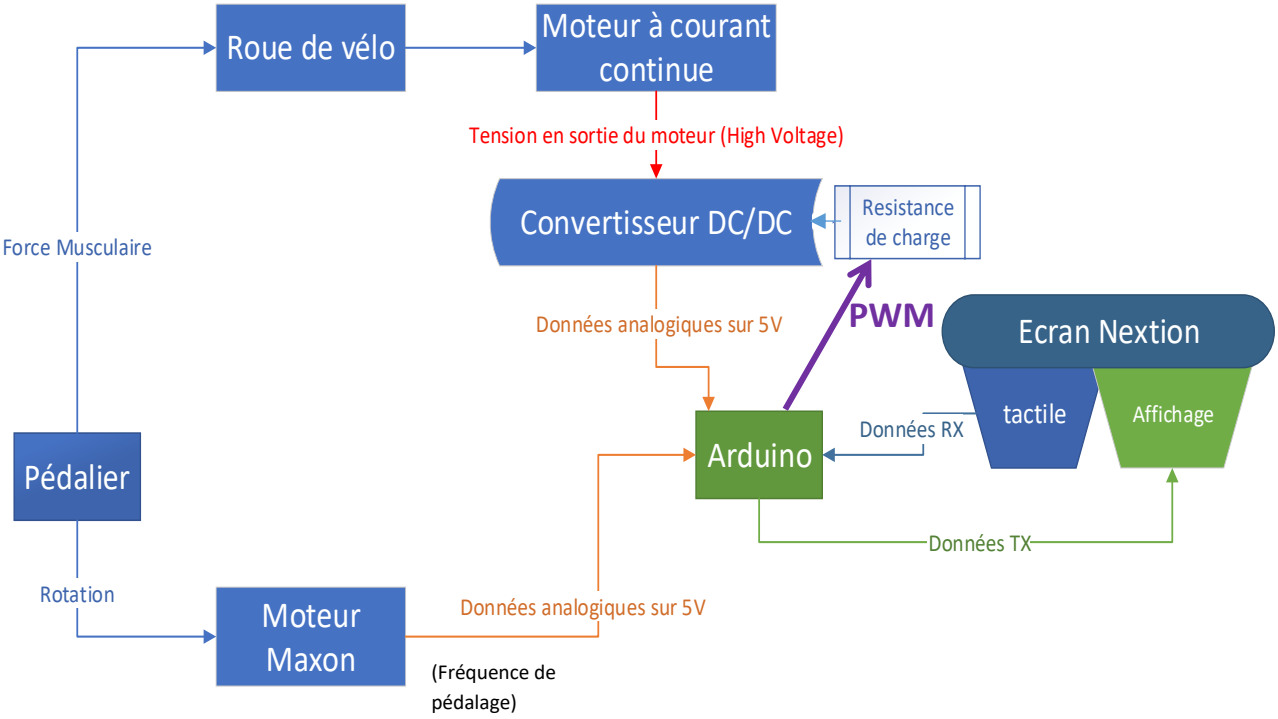


Schéma synoptique :



Fonctionnement :

L'utilisateur agit sur le pédalier, cela produit une puissance mécanique transmise par la chaîne à l'alternateur.

L'alternateur triphasé va produire un courant qui sera ensuite converti par un pont redresseur.

A la sortie du pont le courant est plus ou moins continu, la variation de tension dépend de la capacité du condensateur, la bobine en série avec la charge lisse le courant.

La diode de roue libre permet d'évacuer le courant lorsque le transistor s'ouvre.

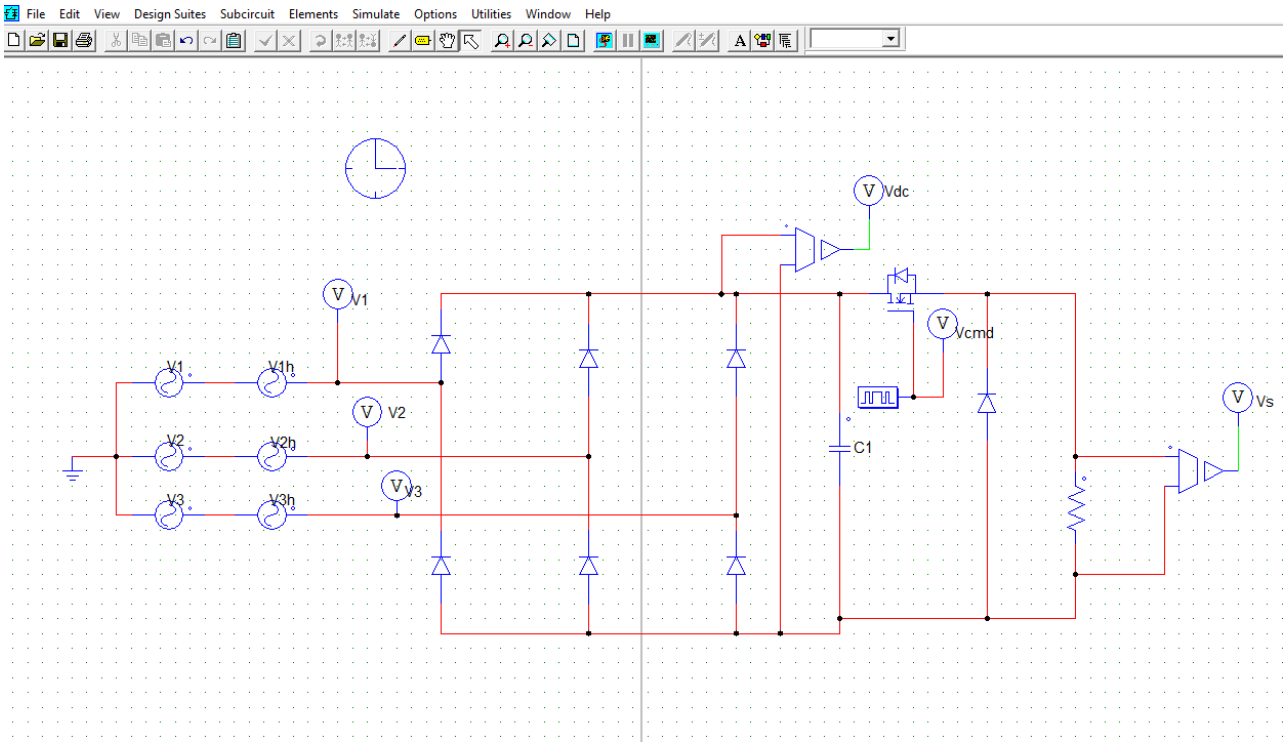
La charge dimensionne le courant.

1ère Partie : L'électronique de puissance.

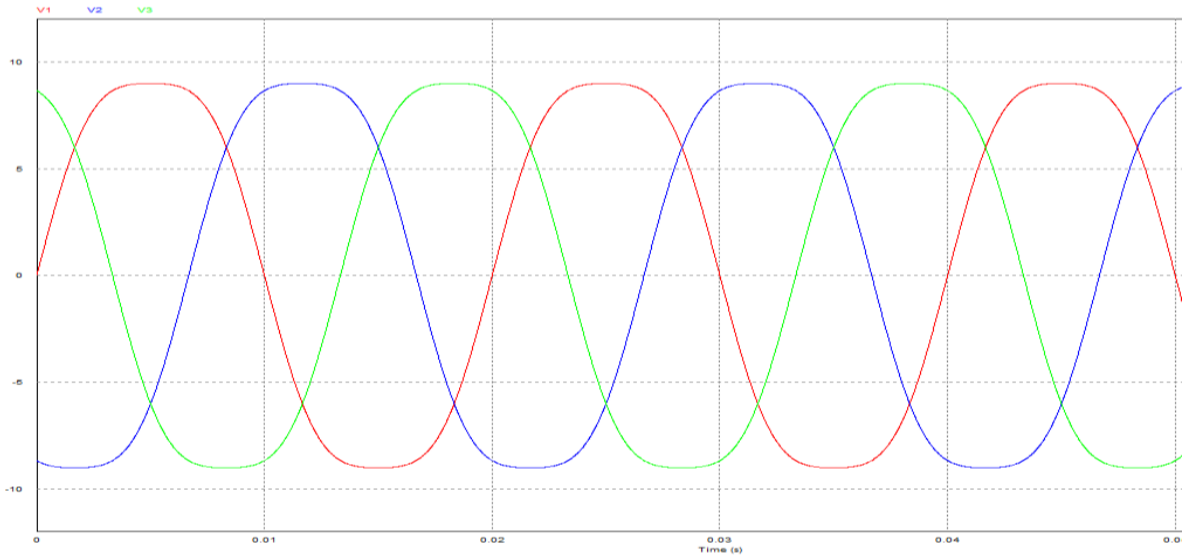
Tout d'abord nous avons commencé par étudier le système, et les documents fournis par notre professeur.

Nous avons utilisé le logiciel PSIM afin de simuler le fonctionnement du moteur triphasé du vélo pour comprendre l'intérêt de l'électronique de puissance. Ce logiciel permet de dessiner le schéma du montage, à partir des éléments de la bibliothèque (machines, transformateurs, interrupteurs électroniques, éléments de commande et de contrôle…). Les appareils de mesure disposés sur le schéma de montage définissent les courbes représentatives des grandeurs électriques et mécaniques que l'on peut obtenir après simulation.

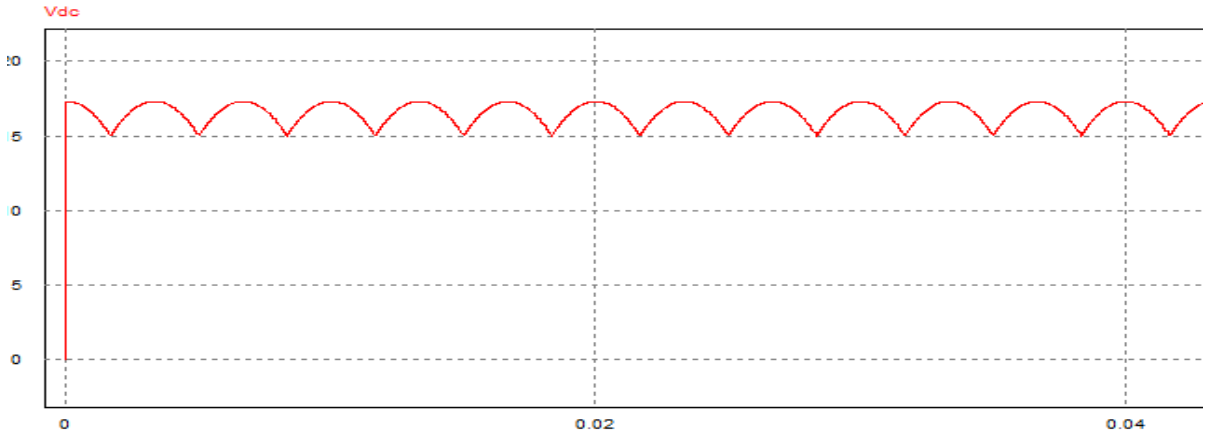
Ici nous allons présenter les trois parties de ce système grâce aux outils de mesure :



- Création d'énergie triphasé.



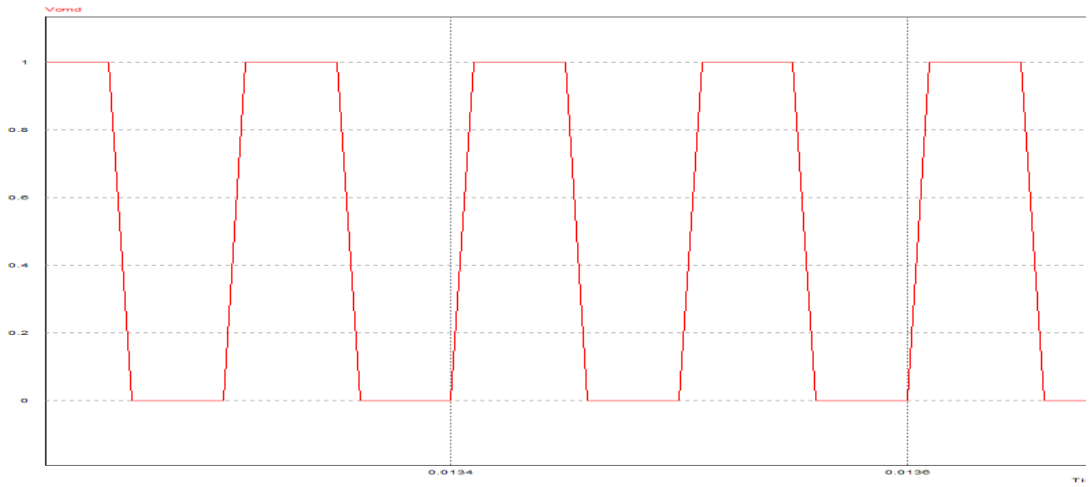
- Le redressement : (convertisseur AC/DC). On constate une tension continue (Vdc).



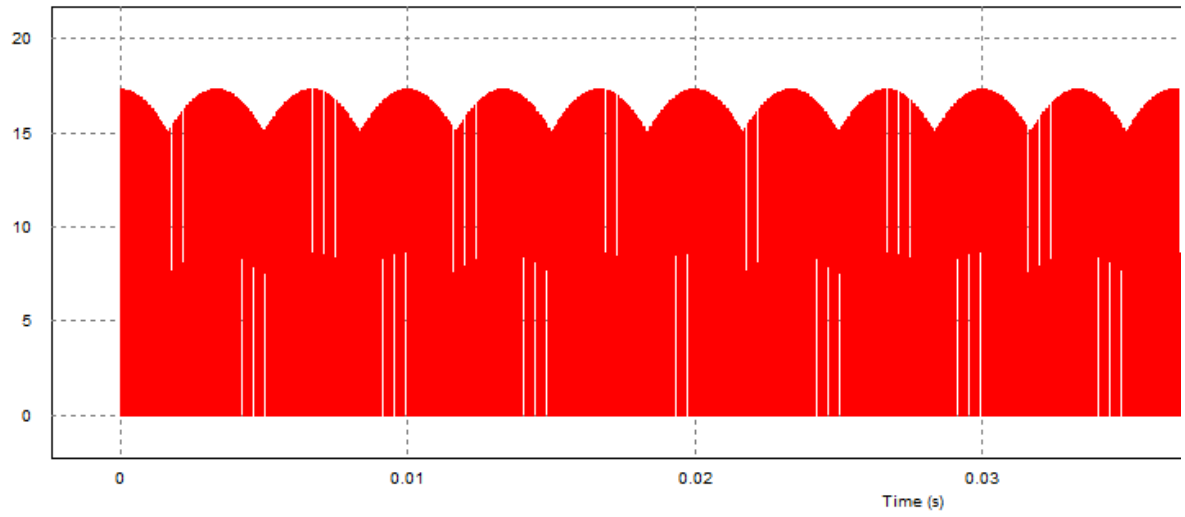
- Le Hacheur série (convertisseur DC/DC).

Le hacheur ou convertisseur continu - continu est un dispositif de l'électronique de puissance mettant en œuvre un ou plusieurs interrupteurs commandés et qui permet de modifier la valeur de la tension d'une source de tension continue avec un rendement élevé. Le découpage se fait à une fréquence très élevée ce qui a pour conséquence de créer une tension moyenne.

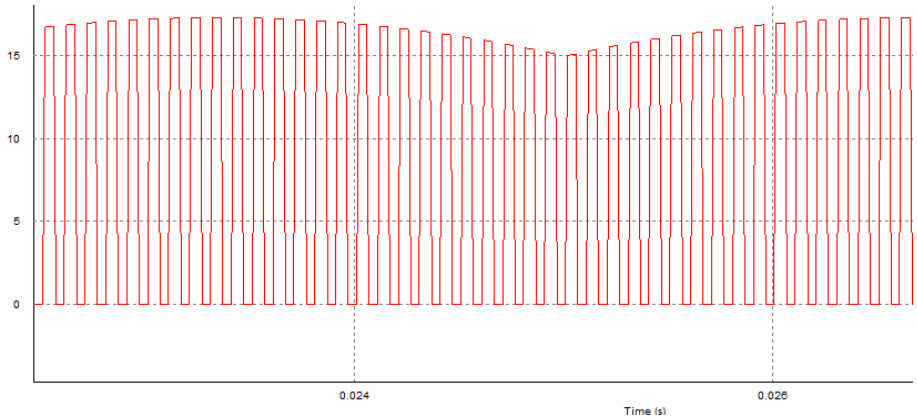
Vcmd : tension de commande qui permet de piloter les interrupteurs.

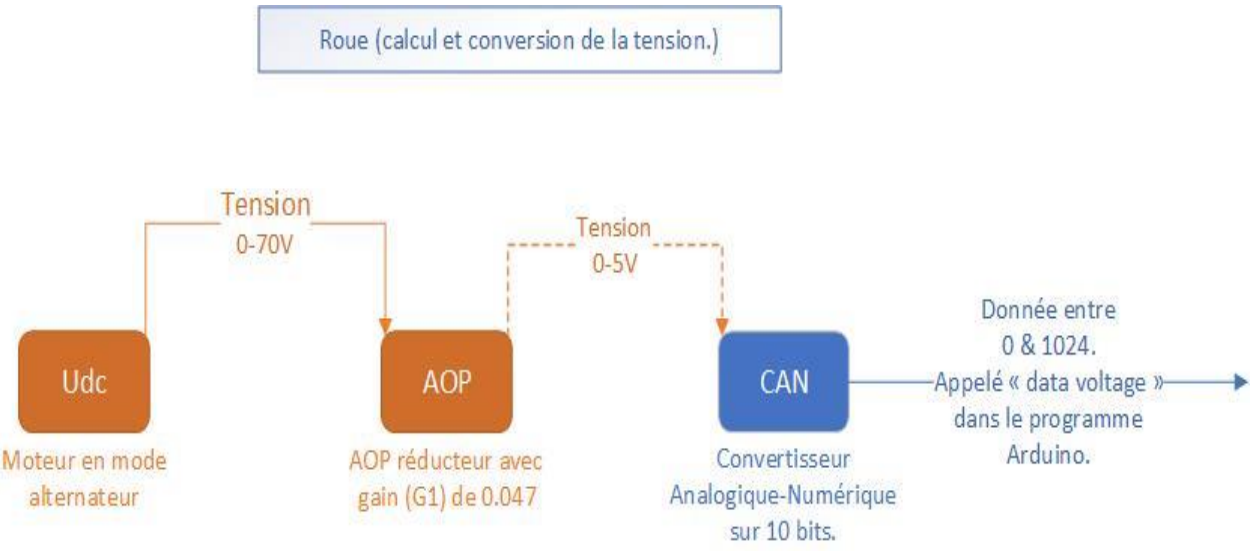


Nous obtenons Vsortie.



Vs zoom :





2ème Partie : L'Interface Homme Machine

Pour répondre au besoin du cahier des charges, il nous a été demandé de réaliser une interface Homme-Machine aussi appelé « IHM » tout en se basant sur un objectif clé : réduire le coût du système. L'ancien système était constitué d'un circuit logique programmable (FPGA) très coûteux pour pouvoir simplement utiliser l'écran de manière correcte. Le fruit de nos recherches nous a permis d'opter pour une solution beaucoup moins coûteuse et tout aussi fonctionnelle : la combinaison Arduino + Ecran Nextion.

- Arduino

Arduino est une carte électronique qui comporte un microcontrôleur programmable. Elle permet de lire des entrées qu'elles soient analogiques ou numériques. Il intègre également plusieurs convertisseurs analogiques-numériques qui nous seront utiles pour notre projet.

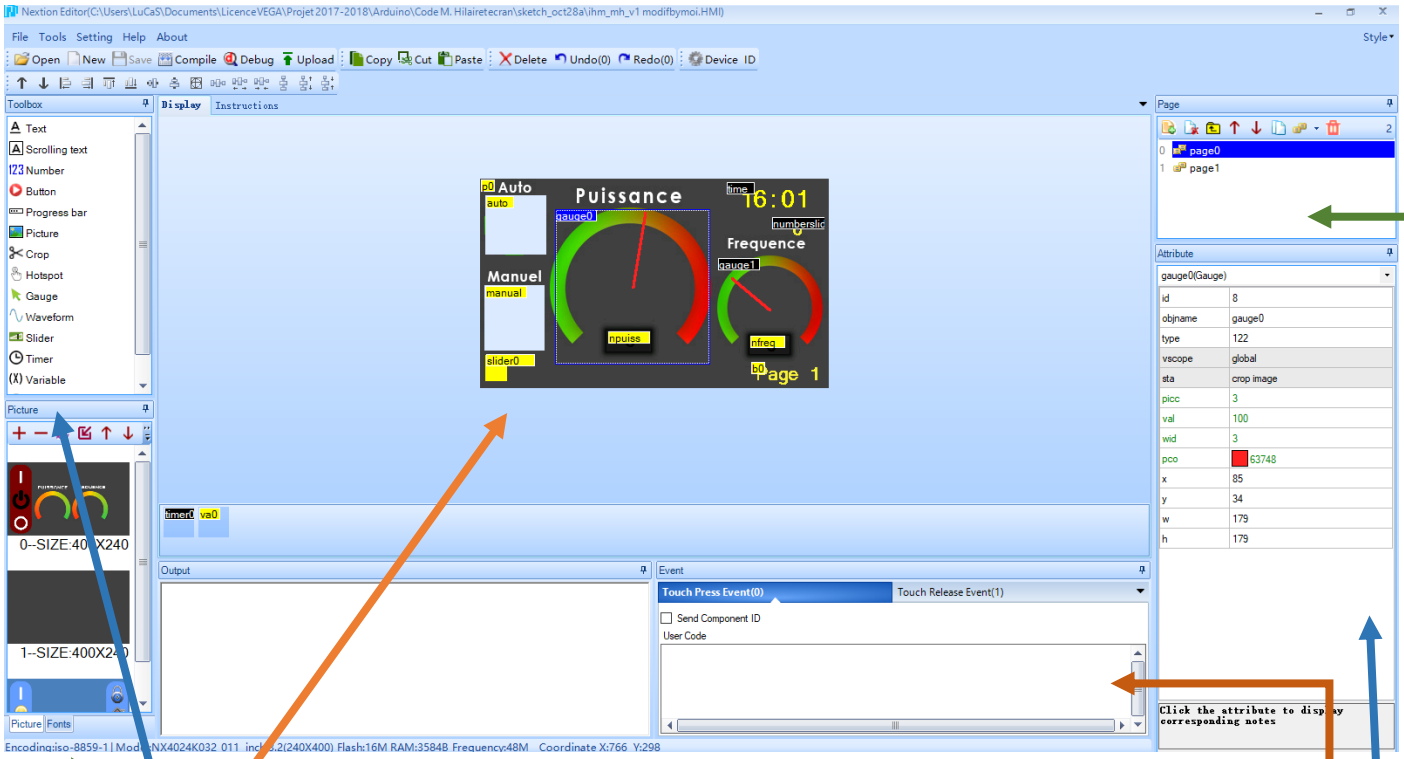
- Nextion

Concrètement l'écran Nextion est un écran tactile qui embarque directement son IHM. Cette IHM est modifiable en intégrant un programme par le biais d'une carte SD ou d'une liaison USB. L'éditeur visuel Nextion Editor est un logiciel WYSIWYG « what you see is what you get ».

Résultat : il n'y a besoin que d'une liaison série pour échanger les données numériques avec un autre équipement (Dans notre cas, l'Arduino).

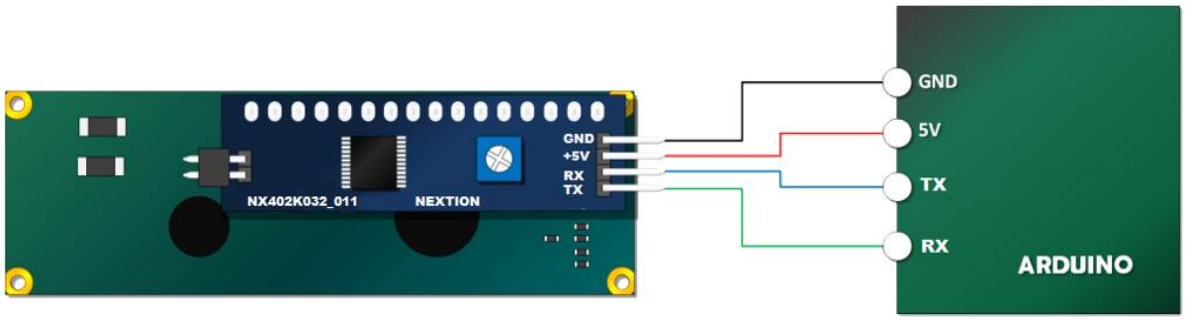
Liaison USB entre Nextion Editor et l'écran Nextion :

Interface Nextion Editor :



- Ici il s'agit de l'écran de visualisation. Le principe du « WYSIWYG » se trouve ici.
- Dans cette fenêtre nous avons toutes les fonctions que « Nextion Editor » propose.
- Il s'agit de la bibliothèque d'images.
- Dans cette fenêtre on peut écrire du code pour par exemple lire la valeur d'une autre fonction et réagir en conséquence.
- Fenêtre dite « paramètres de la fonction » on peut modifier le nom de la fonction, sa couleur, son identifiant etc…
- Enfin, cette fenêtre sert à la gestion des différentes pages que l'on peut afficher sur l'écran.

Voici la liaison (simple) de la connexion entre l'écran Nextion et l'Arduino.



La position des aiguilles sur Nextion.

Nous devons intégrer une jauge avec une aiguille pour pouvoir afficher la puissance. Nous avons donc utilisé la fonction « gauge » qui est prévu à cet effet. Il s'agit d'une aiguille capable de bouger sur 360° en fonction d'une valeur donnée.

Un des problèmes rencontrés a été que lorsqu'on donne la valeur « 0 » à l'aiguille, l'aiguille commence sa course à -90°. Pour résoudre ce problème nous avons trouvé une méthode directement depuis le programme en langage « C ». Nous avons pour cela créé un tableau :

```
36 int mygVals[55] = {316, 321, 326, 331, 336, 341, 346, 351, 356, 1, 6, 11, 16, 21, 26,
37 31, 36, 41, 46, 51, 56, 61, 66, 71, 76, 81, 86, 91, 96, 101, 106,
38 111, 116, 121, 126, 131, 136, 141, 146, 151, 156, 161, 166, 171,
39 176, 181, 186, 191, 196, 201, 206, 211, 216, 221, 226};
```

Grâce à ce tableau nous allons assimiler la valeur « 0 » venant de notre moteur à la première valeur de notre tableau en l'occurrence « 316 ». L'aiguille sera donc à la valeur « 0 » et en position 316° comme voulu. Cette même méthode est appliquée pour toutes les autres valeurs. On incrémente de 5 entre chaque valeur.

Les librairies.

Nous avons ajouté les deux librairies (SoftwareSerial.h & Nextion.h (500Mb) dans le logiciel Arduino pour pouvoir intégrer l'écran Nextion et nous avons dû modifier le fichier « NexConfig.h » de la librairie Nextion. Il a fallu modifier ce fichier en l'ouvrant avec un traitement de texte simple comme celui fournit avec Windows. Il faut ensuite modifier une ligne pour ensuite la remplacer par trois autres lignes. Cette manipulation est disponible, voir bibliographie.

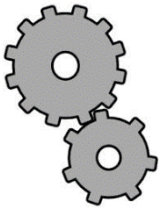
La partie affichage de l'écran permet d'informer en temps réel la puissance développée par l'utilisateur.

La partie tactile permet de régler en temps réel la résistance du moteur afin de simuler une pente selon les envies de l'utilisateur, celle-ci est une charge que l'on va appliquer au moteur à l'aide d'un potentiomètre (ici le potentiomètre sera pilotable depuis l'écran tactile).

Stratégie utilisée pour déterminer la cadence de pédalage :

Notre Home trainer est équipé sur son pédalier d'un moteur Maxon A-MAX 110150 inutilisé. (Doc en annexe)

C'est un moteur pas à pas, à courant continue et équipé d'aimants permanents.



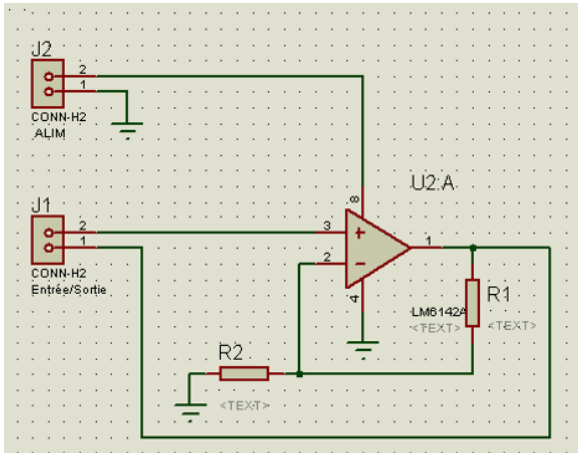
Le moteur est relié au pédalier par deux engrenages.

Un calcul qui divise le nombre total de cran des deux engrenages permet de trouver le coefficient de réduction : ici nous avons 60/14=4.285

D'après la documentation, nous avons trouvé qu'en sortie de ce moteur, nous obtenons une tension inférieure à 1V.

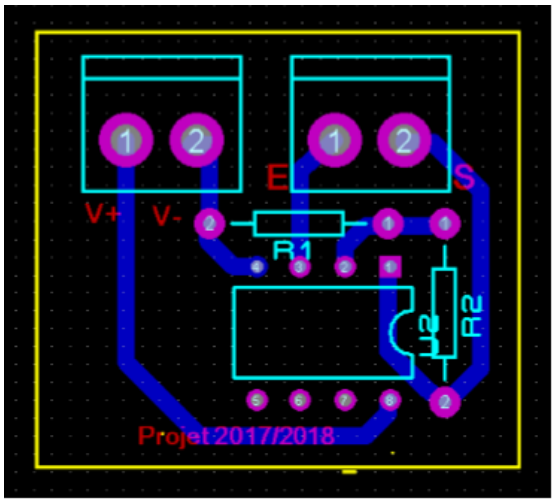
Nous avons donc besoin d'amplifier cette valeur à la valeur réelle pour qu'elle soit utilisable, en utilisant un Amplificateur non-inverseur.

Nous avons utilisé le logiciel ISIS de Proteus pour éditer le schéma électrique.

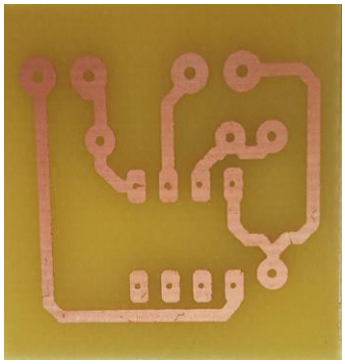


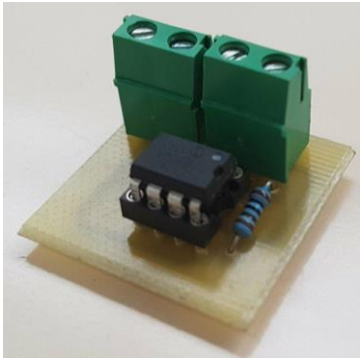
La même masse est utilisée pour alimenter l'AOP et Ve.

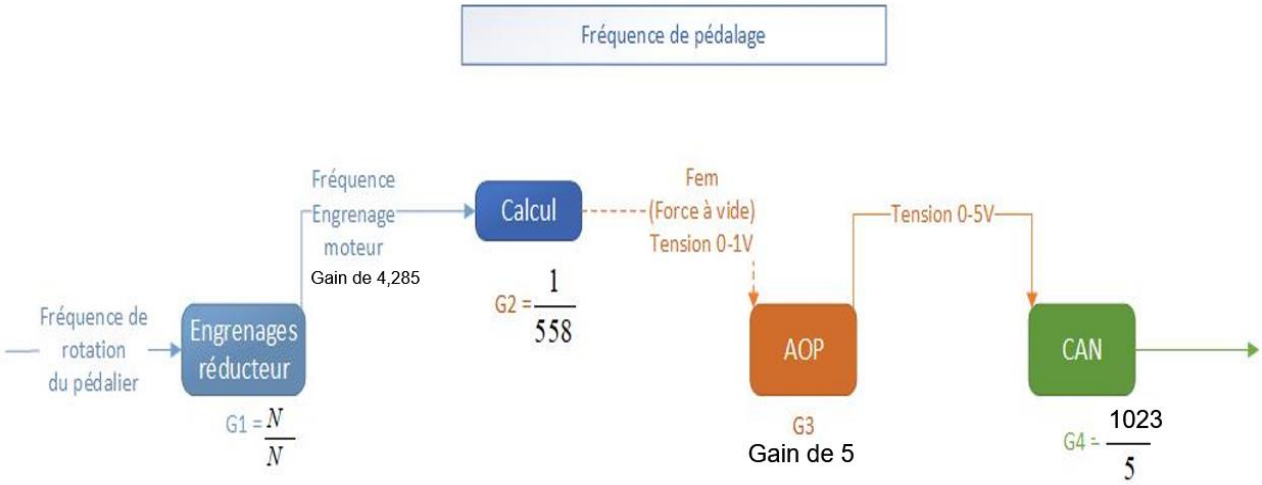
Le logiciel ARES pour l'édition et de routage du circuit électronique :



Nous avons percé la carte et brasé les composant que nous avions préalablement choisi en effectuant des tests sur une plaque Labdec.







L'AOP est supposé idéal, en régime linéaire (V+ = V-).

Le courant d'entrée sur la broche inverseuse étant nul, on peut appliquer un pont diviseur de tension résistif entre **R1** et **R2** pour obtenir la tension V- :

$$V_- = V_s\left(\frac{R_2}{R_1+R_2}\right)$$

$$V_+ = V_e$$

$$\Rightarrow V_e = V_s\left(\frac{R_2}{R_1+R_2}\right)$$

$$\Rightarrow V_s = V_e\left(1+\frac{R_1}{R_2}\right)$$

La tension de sortie Vs est donc bien supérieur ou égale à la tension d'entrée Ve (si R1/R2 <<1), et de même signe, d'où son appellation amplificateur non-inverseur.

C'est donc bien ce dont nous avons besoin, un montage à gain réglable.

Avec : R2 = 1kΩ ; R1 = 250Ω nous obtenons un coefficient d'amplification entre Ve et Vs ≈ 5.

Nous avons mesuré que la tension de sortie du moteur à vide (FEM[1] en volt) est entre 0-1V, avec un coefficient d'amplification de 5 nous obtenons une valeur exploitable par l'Arduino.

[1] Force électromotrice

Conclusion :

Ce projet s'est révélé très formateur dans la mesure où il a consisté en une approche concrète de notre futur métier. En effet, la prise d'initiative, le respect des délais et le travail en équipe sont des aspects essentiels.

Nous avons cependant accumulé un retard dans la compréhension du travail effectué précédemment, ainsi que dans la plus grande partie du projet que nous avions sous-estimée. La programmation, dû à la complexité des actions à exécuter dans les règles, mais après un certain temps de réflexion, nous sommes parvenues à résoudre nos problèmes.

Grâce à ce projet, nous avons pu appliquer et approfondir nos connaissances obtenues par les différentes formations suivies cette année dans le domaine de l'électronique de puissance, la gestion de projet, l'informatique embarquée (langage C), la conception assistée par ordinateur, ainsi que du traitement du signal.

Bibliographie :

Librairie Arduino :

https://create.arduino.cc/projecthub/tsavascii/nextion-lcd-communicate-with-arduino-uno-188a44

Datasheet AOP fréquence de pédalage :

http://ww1.microchip.com/downloads/en/DeviceDoc/21668d.pdf

Aide Nextion Editor :

https://www.itead.cc/wiki/Nextion_Editor_Quick_Start_Guide

https://www.itead.cc/blog/an-entry-application-%20of-progress-bar-picture-and-crop-component

ANNEXES :

Programmation :

Final_08_03_1

```
#include <SoftwareSerial.h>            //Librairie de SoftWareSerial.h
#include "Nextion.h"                   //Librairie de Nextion.h

// Config port serie
SoftwareSerial HMISerial(10,11);       //Définition des pins RX, TX.

#define DEBUG

#define gain  0.194    // 199/1023 => "OCR2A/valeur max." CAN gain d'adaptation.
#define gain2 0.2490   // 255/1024 => gain d'adaptation.



// Declaration des composants
// [page id:0,component id:3, component name: "n0"].
NexText time  = NexText(0, 7, "time");                  // N° de page, N° d'objet, Nom de l'objet ?
NexCrop light = NexCrop(0, 3, "light");
NexHotspot on_light  = NexHotspot(0, 8, "on");
NexHotspot off_light = NexHotspot(0, 4, "off");

//Declaration du potentiomètre en bas de l'écran pour régler la pente.
NexText   slider0_text = NexText(0, 13, "slider0_text");
NexSlider slider0 = NexSlider(0, 12, "slider0");

//Declaration de l'aiguille.
NexTimer timer0 = NexTimer(0, 11, "timer0");
NexGauge gauge0  = NexGauge(0, 1, "gauge0");

 NexTouch *nex_listen_list[] =

{
  &on_light,
  &off_light,
  &slider0,
//  &timer0,
  NULL
};



//Declaration des variables.
unsigned short int data_voltage;               //variable pour la données du moteur en Watts(0,1023)
unsigned short int data_potentiometer;         //variables pour la donnée de la pente(0,1023)

unsigned char test = 0;

char stime[6];
unsigned long int t, t_nex;
uint32_t number = 0;                           //Déclaration de "number" à "0"
unsigned int count = 1;                        // Déclaration de "count" à "1"            --
int upperLimit = 54;                           // Limite haute - myVals[] upper dimension  --
int lowerLimit = 0;                            // Limite basse                             --
char buffer[100] = {0};                        //Définition du buffer à "0" le
```

```
// on incremente de 5 - début de la jauge = 316, fin de la jauge = 226
//mygVals[55] est un tableau de 55 caractères.
int mygVals[55] = {316, 321, 326, 331, 336, 341, 346, 351, 356, 1, 6, 11, 16, 21, 26,
31, 36, 41, 46, 51, 56, 61, 66, 71, 76, 81, 86, 91, 96, 101, 106,
111, 116, 121, 126, 131, 136, 141, 146, 151, 156, 161, 166, 171,
176, 181, 186, 191, 196, 201, 206, 211, 216, 221, 226};


void Slider0PopCallback(void *ptr)
{
  uint32_t val = 0;
  char temp[10] = {0};

  slider0.getValue(&val);       // on prend la valeur du slider et on l'attribu a "val"
  utoa(val, temp, 10);          //fonction à revoir
  slider0_text.setText(temp);
  gauge0.setValue(val);         // on définit la valeur de "gauge" par la variable "val"

  //Affichage de "slider0 = *valeur*"
  Serial.print("slider0 = ");
  Serial.println(val);
}
```

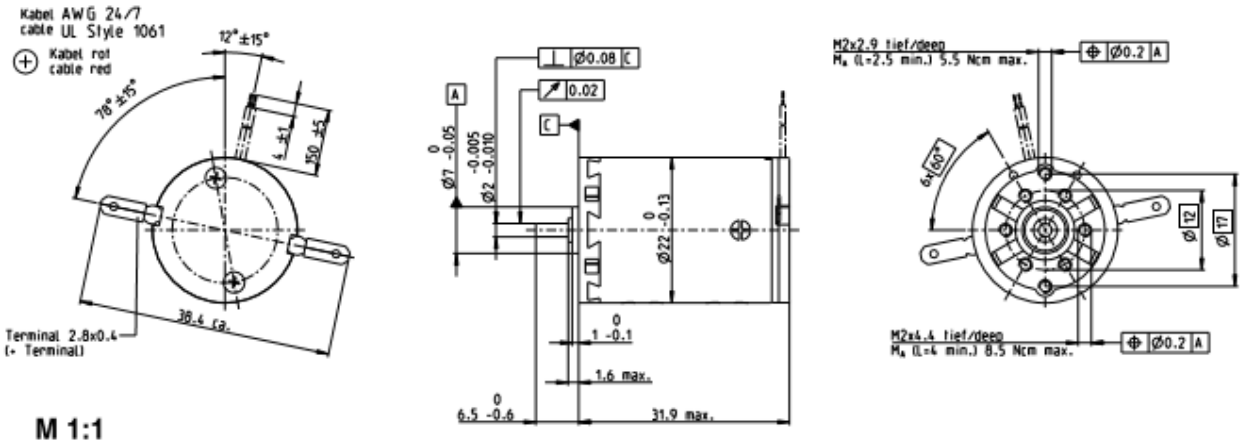
```
//------------------------------------------------------------------------------
void setup()
{
  // Configuration des entrées analogiques
  pinMode(A0, INPUT);  // bus de tension continue
  pinMode(A1, INPUT);  // sortie de potentiometre

  // Config Pin 8 for the frequency measurement
  pinMode(8,INPUT); // Capteur avec fréquence de la roue
  pinMode(2,INPUT); // Switch

  // Config port serie pour communiquer avec le PC (debug)
  #ifdef DEBUG
    Serial.begin(9600);                  // initialisation de la communication série à 9600 bps
  #endif

  //------------------------------------------------------------------------------
  noInterrupts();           // desactivation des interruptions

  //------------------------------------------------------------------------------
  // PWM configuration
  pinMode( 3, OUTPUT);

  // Timer 2 (8 bists)
  // page 153
  // 00=Normal port operation, OC2A disconnected,
  // 10=Clear OC2B on Compare Match, set OC2B at BOTTOM, (non-inverting mode).
  // --
  // WGM21=1, WGM20=1
  TCCR2A = 0b00100011;

  // page 155, Fast PWM, mode = 7, TOP = OCRA
  // 00= for a non-PWM mode
  // --
  // 1=WGM22
```

```
138   // 010=clk/8 (From prescaler) CS22=0, CS21=1, CS20=0
139   TCCR2B = 0b00001010;
140
141   // Page 141, The PRTIM2 bit must be written to zero to enable
142   // Timer/Counter2 module.
143   PRR = PRR & 0b10111111;
144   // Page 157, Timer/Counter Register
145   TCNT2 = 0;
146
147   // 100.10-6 = (OCR2A+1)*1/(16MHz/8)
148   OCR2A = 199;
149   OCR2B = 100; // Duty cycle
150
151   // Interrupt configuration
152   // Page 158 : Enable TIMER2 interrupt when TCNT2 = OCR2A
153   TIMSK2 = 0x02;
154
155   //-------------------------------------------------------------------------
156   // TIMER1 configuration (16 bits)
157   // Page 131
158   // 0000= Normal port operation, OC1A/OC1B disconnected.
159   // --
160   // 00=WGM11 and WGM10, mode = 4, page 132
161   TCCR1A = 0b00000000;
162   // ---
163   // 01= WGM13=1 and WGM12=1, mode = 4, page 134
164   // 101=CS12,CS11,CS10=clk/1024
165   // 011 /64
166   // 100 /256
167   TCCR1B = 0b00001100;  // mode 4
168
169   // Page 134, Timer/Counter Register
170   TCNT1H = 0;
171   TCNT1L = 0;
173   // TOP value of TCNT
174   OCR1AH = 0xF3;
175   OCR1AL = 0x24;
176
177   // Interrupt configuration
178   TIMSK1 = 0x02;
179
180   //-------------------------------------------------------------------------//**
181   interrupts();        // activation des interruptions
182
183     /* Set the baudrate which is for debug and communicate with Nextion screen. */
184   nexInit();
185   //on_light.attachPush(OnLightPushCallback, &on_light);
186   //off_light.attachPush(OffLightPushCallback, &off_light);
187   on_light.attachPush(OnLightPushCallback);
188   off_light.attachPush(OffLightPushCallback);
189   slider0.attachPop(Slider0PopCallback);
190 //  timer0.attachTimer(Timer0Callback);
191   dbSerialPrintln("setup done");
192 }
193
```

```
196 void loop()                                    //boucle
197 {
198    nexLoop(nex_listen_list);
199
200   // Lire la tension a l'entrée du convertisseur DC/DC
201   data_voltage = analogRead(A0); // Lecture de la pin A0
202 //  data_voltage_send = (unsigned char)(gain * data_voltage);  // Entre 0 et 199=OCR2A
203
204   mode = digitalRead(2);  // State of the switch
205 }
208 // ISR TIMER2 - 100µs
209 ISR(TIMER2_COMPA_vect)
210 {
211
212 }
213 //-------------------------------------------------------------------------------
214 // ISR TIMER1 - 1s
215 ISR(TIMER1_COMPA_vect)
216 {
217   // Affichage de : voltage
218   #ifdef DEBUG
219     Serial.print("data_voltage=");
220     Serial.println(data_voltage);
221   #endif
222
223   // Lecture puis affichage du potentiomètre
224   data_potentiometer = analogRead(A1); // Lecture de la pin A1
225   #ifdef DEBUG
226     Serial.print("data_potentiometer=");
227     Serial.println(data_potentiometer);
228   #endif
229
230   // Choisir entre le potentiomètre physique ou le slider de l'écran tactile
231   // Duty cycle
232   // La valeur Max de OCR2B est OCR2A=199
233   if(mode == 1)
234   {
235     //partie potentiomètre
236     OCR2B = (unsigned char)(data_potentiometer * gain);  //valeur final du potentiomètre envoyé à l'Arduir
237     #ifdef DEBUG
238       Serial.println("Control via the potentiometer");
239     #endif
240   }
243   //partie ecran tactile
244   else
245   {
246 //OCR2B = data_usb * 2;  // Max = 199 = 2*100%
247
248     #ifdef DEBUG
249       Serial.println("Control via the touchscreen");
250       Serial.print("data=");
251 //    Serial.println(data_usb);
252     #endif
253     gauge0.setValue(mygVals[count]);                    //On attribue une valeur du tableau à l'aiguille.
254   }
255
256 }
```

Moteur Maxon A-MAX 110150 :

A-max 22 ∅22 mm, Graphite Brushes, 6 Watt



M 1:1

Stock program
Standard program
Special program (on request)

Part Numbers

with terminals	110143	110145	110146	**110147**	110148	110149	110150	**110151**	110152	110153	110154	110155	
with cables	139840	353017	199807	320206	323856	108828	199424	202921	267433	325492	313302	353019	

Motor Data														
Values at nominal voltage														
1	Nominal voltage	V	6	9	9	12	12	15	18	24	24	36	48	48
2	No load speed	rpm	9240	9690	8500	10200	9170	10000	9770	10500	8480	9630	9110	8210
3	No load current	mA	83.1	57.9	49.6	45.8	40.5	36	29	23.7	18.4	14.2	9.99	8.84
4	Nominal speed	rpm	6240	6530	5350	7060	6000	6890	6600	7380	5270	6420	5840	4940
5	Nominal torque (max. continuous torque)	mNm	5.91	6.88	7.04	6.96	6.95	6.93	6.92	6.9	6.97	6.86	6.75	6.86
6	Nominal current (max. continuous current)	A	1.08	0.859	0.77	0.681	0.613	0.534	0.432	0.347	0.283	0.21	0.147	0.135
7	Stall torque	mNm	19.4	22.1	19.8	23.7	20.9	22.9	22	23.7	18.9	21.1	19.2	17.6
8	Stall current	A	3.29	2.59	2.04	2.17	1.72	1.65	1.29	1.12	0.721	0.606	0.393	0.325
9	Max. efficiency	%	67	70	69	72	70	72	72	73	70	72	71	70
Characteristics														
10	Terminal resistance	Ω	1.82	3.48	4.42	5.53	6.96	9.09	14	21.5	33.3	59.4	122	148
11	Terminal inductance	mH	0.106	0.223	0.288	0.363	0.445	0.585	0.891	1.37	2.1	3.69	7.3	8.97
12	Torque constant	mNm/A	5.9	8.55	9.73	10.9	12.1	13.9	17.1	21.2	26.2	34.8	48.9	54.3
13	Speed constant	rpm/V	1620	1120	981	875	790	689	558	450	364	274	195	176
14	Speed / torque gradient	rpm/mNm	500	454	446	444	455	452	457	456	461	468	487	479
15	Mechanical time constant	ms	20.9	20.2	20.1	19.9	19.9	19.9	19.7	19.7	19.8	19.7	19.9	19.8
16	Rotor inertia	gcm²	4	4.25	4.3	4.29	4.19	4.2	4.13	4.13	4.09	4.02	3.9	3.94

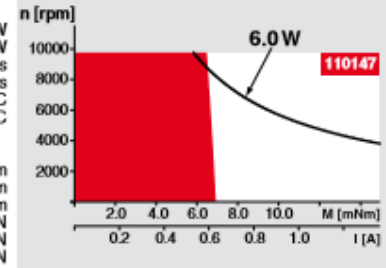
Specifications

Thermal data		
17	Thermal resistance housing-ambient	20 K/W
18	Thermal resistance winding-housing	6.0 K/W
19	Thermal time constant winding	9.43 s
20	Thermal time constant motor	314 s
21	Ambient temperature	-30...+85°C
22	Max. winding temperature	+125°C
Mechanical data (sleeve bearings)		
23	Max. speed	9800 rpm
24	Axial play	0.05 - 0.15 mm
25	Radial play	0.012 mm
26	Max. axial load (dynamic)	1 N
27	Max. force for press fits (static)	80 N
28	Max. radial load, 5 mm from flange	2.8 N
Mechanical data (ball bearings)		
23	Max. speed	9800 rpm
24	Axial play	0.05 - 0.15 mm
25	Radial play	0.025 mm
26	Max. axial load (dynamic)	3.3 N
27	Max. force for press fits (static)	45 N
28	Max. radial load, 5 mm from flange	12.3 N
Other specifications		
29	Number of pole pairs	1
30	Number of commutator segments	9
31	Weight of motor	54 g

Values listed in the table are nominal.
Explanation of the figures on page 79.

Option
Ball bearings in place of sleeve bearings

Operating Range



Comments

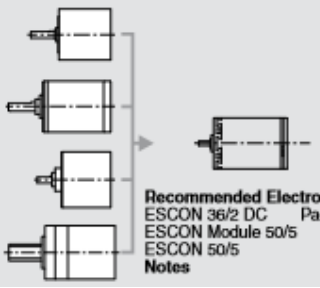
Continuous operation
In observation of above listed thermal resistance (lines 17 and 18) the maximum permissible winding temperature will be reached during continuous operation at 25°C ambient.
= Thermal limit.

Short term operation
The motor may be briefly overloaded (recurring).

Assigned power rating

maxon Modular System Overview on page 20–25

Planetary Gearhead
∅22 mm
0.1 - 0.6 Nm
Page 260/261
Planetary Gearhead
∅22 mm
0.5 - 2.0 Nm
Page 262/264
Spur Gearhead
∅24 mm
0.1 Nm
Page 269
Spindle Drive
∅22 mm
Page 299/300



Recommended Electronics:

ESCON 36/2 DC	Page 342
ESCON Module 50/5	343
ESCON 50/5	344
Notes	22