Solex électrique

Alain Bertout et Philippe Loutrel

F6KFA, janvier 2022 - mai 2022

Le Solex à essence

Le rappel de l'existant

Le VeloSolex était un moyen de transport qui se situait entre le vélo et le cyclomoteur. Il s'agissait un véhicule à traction avant, en effet, il était équipé d'un moteur thermique de 49cm^3 qui faisait tourner un petit galet cylindrique entrainant la roue avant.

Un VeloSolex modèle 2200 se trouvait disponible pour des essais dans le garage. Le Solex 2200 est sorti en juin 1961, le moteur avait une puissance atteignant 0,7 cheval (500 Watt), il était équipé d'un levier de relevage du moteur et d'une manette de décompression pour faciliter le démarrage.

La bicyclette qui roule et monte toute seule

Le lancement du Solex nécessitait de le pousser pour démarrer le moteur et de pédaler ensuite sur une dizaine de mètre pour aider le moteur à atteindre son plein régime.

Le Solex 2200 avait une consommation de 1,4 litres/100 km avec un mélange d'essence 2 Temps pour une autonomie de 100 km, soit 3 h à 30km/h.

Le Solex électrique

Les origines du projet

Le club scientifique et radio amateur F6FKA a reçu plusieurs demandes de réparation de trottinettes électriques. Ces trottinettes étaient équipées de moteur sans balais (« brushless ») nécessitant des alimentations alternatives triphasées à fréquences variables.

Il fut rapidement trouvé que les pannes provenaient des 6 transistors MOSFET se trouvant dans les contrôleurs de moteurs. Le coût de ces transistors étant de 2 € pièce, la réparation n'était pas rentable car un contrôleur neuf livré en France coutait entre 10 et 15 €. (4.5€ livré en Chine).

Un contrôleur de moteur « brushless » 36V

Les explications sur le système : batterie, contrôleur, moteur

Le club a cherché à comprendre comment fonctionnait ce contrôleur de moteur.

Le contrôleur doit produire trois tensions alternatives d'amplitude et de fréquence variables à partir d'une tension d'entrée continue de l'ordre de 36V. Les batteries sont habituellement de type au lithium avec une charge de l'ordre de 10Ah. Les puissances moteur sont de 250W pour une trottinette d'entrée de gamme ou un vélo à assistance électrique, de 350W pour les hoverboards. On trouve cependant

des contrôleurs allant jusqu'à 1000W fonctionnant avec des batteries plus puissantes de 48V (hors législation française). Un système de limitation de courant est nécessairement mis en œuvre dans le contrôleur afin d'éviter de consommer plus de 10A et de préserver ainsi la durée de vie de la batterie au lithium. Il faut normalement travailler avec des courants moyens proches de la charge (=10A pour 10Ah).

Les moteurs « brushless » sont le plus souvent équipés de trois capteurs Hall pour informer le contrôleur de la position courante de l'arbre du moteur par rapport au bobinage et aux aimants permanents. Le fonctionnement des moteurs « brushless » est réversible car ils peuvent servir d'alternateur et permettre la récupération d'énergie électrique lors de décélérations et les descentes.

La découverte du fonctionnement du système.

Au club F6KFA, plusieurs membres ont fait l'acquisition de roues/moteurs 200/50 d'hoverboard d'occasion (34,5€ pièce).

La roue/moteur 200/50 à pneu plein pour hoverboard

Les caractéristiques des roues choisies sont:

- Une puissance nominale consommée de 350W pour une alimentation de 36V.
- Une période de l'onde électrique correspond à 24° de rotation du moteur.
- Un câblage interne du moteur prévoyant 15 groupes de bobines mises en parallèle et réparties sur le pourtour de l'axe.
- Un rotor comportant les aimants permanents.

Un tour de roue s'exécute donc en 15 rotations élémentaires de 24° chacune. Le diamètre de la roue est de 19 cm avec circonférence de 61 cm.

Les enroulements des bobines des phases ont une résistance R de 0,25Ω et une inductance L de 0.3mH. Entre deux phases en étoile la résistance est ainsi de 0,5Ω et l'inductance de 0.7mH.

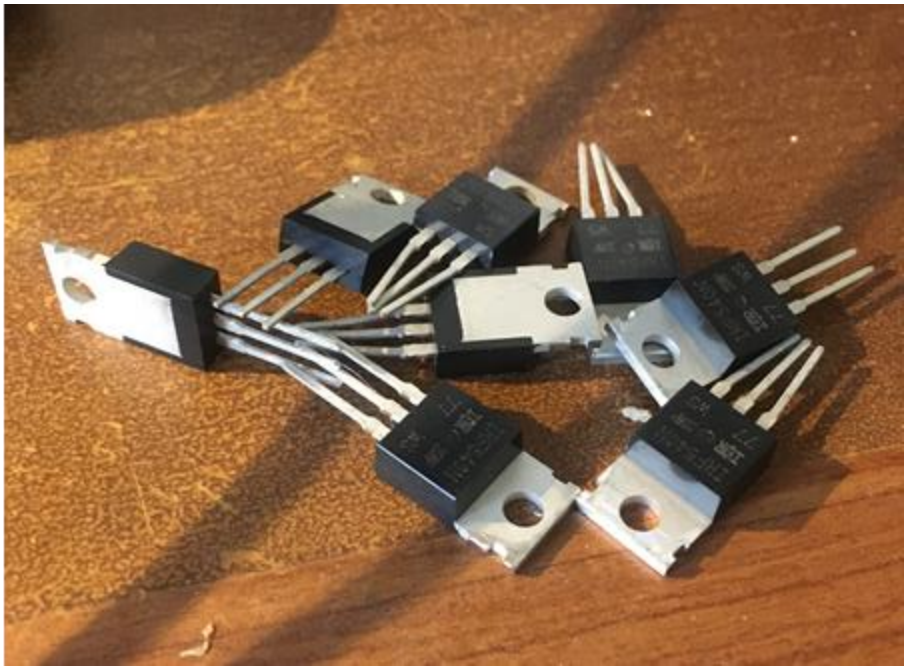
Les premiers tests

Des premières maquettes de contrôleur ont été réalisées pour produire des tensions triphasées d'amplitudes variables (< 12V) et de fréquences variables (< 50Hz). Les informations provenant des capteurs Hall du moteur n'étaient alors pas prises en compte.

La vitesse du moteur suivait bien la fréquence, mais à certaines vitesses, la consommation montait rapidement et le moteur devenait bruyant. Un réglage délicat de la tension en fonction de la fréquence permettait de réduire le bruit mais souvent au détriment de la puissance mécanique du moteur.

Il a fallu changer de principe et construire des nouvelles maquettes pour tenir compte des informations des capteurs Hall et ne plus forcer la fréquence et laisser le moteur trouver sa fréquence tout seul. La vitesse du moteur se réglait alors par la tension d'alimentation et la puissance mécanique consommé sur la roue-moteur.

Plusieurs montages de contrôleurs ont été conçus et réalisés (commutation par relais, par transistors IGBT, et par MOSFET). Les courants appelés au démarrage étaient très élevés (20V / 0,5Ω = 40A). Les alimentations régulées devaient facilement limiter cette pointe de courant. Les transistors MOSFET utilisés étaient des IRF540 ne supportant en saturation que des courants de 33A, de nombreux claquages furent constatés en particulier lors des raccordements à des batteries qui, elles, n'offrent pas de limitation de courant !



Quelques transistors MOSFET IRF540 claqués en courant

Plusieurs principes de commandes de grille des transistors MOSFET furent étudiés : à base de coupleurs optoélectroniques (pas toujours assez rapides et travaillant à basse tension < 25V) puis à l'aide de circuits drivers IR2112 (plus rapides et travaillant pour des tensions plus élevés (de 10 à 600 Volts …).

Solutions retenue pour le contrôleur

Chaque circuit IR2112 pilote une paire de transistor MOSFET avec des commandes de grille Vgs de 12V.

Un microcontrôleur Arduino Nano héberge les fonctions suivantes :

- Lecture des trois capteurs Hall du moteur (fils Jaune, Bleu, Vert
- Commandes séquentielles des transistors (alternativement pour chaque phase, blocage ou saturation du transistor du haut ou du transistor du bas)
- Lectures de la consigne de vitesse (lecture d'une manette d'accélérateur à effet Hall
- Lecture d'un capteur de courants ACS712 (pour effectuer une limitation de courant à 10A)
- Commande de tension variable par Modulation de Largeur d'Impulsion (MLI ou PWM)

Les sorties des trois phases moteur correspondent aux couleurs des trois capteurs Hall: Jaune, Bleu, Vert. Les phases sont tour à tour connectée au +36V ou au GND ou restée flottante (non alimentée). A chaque phase il y a donc un couple

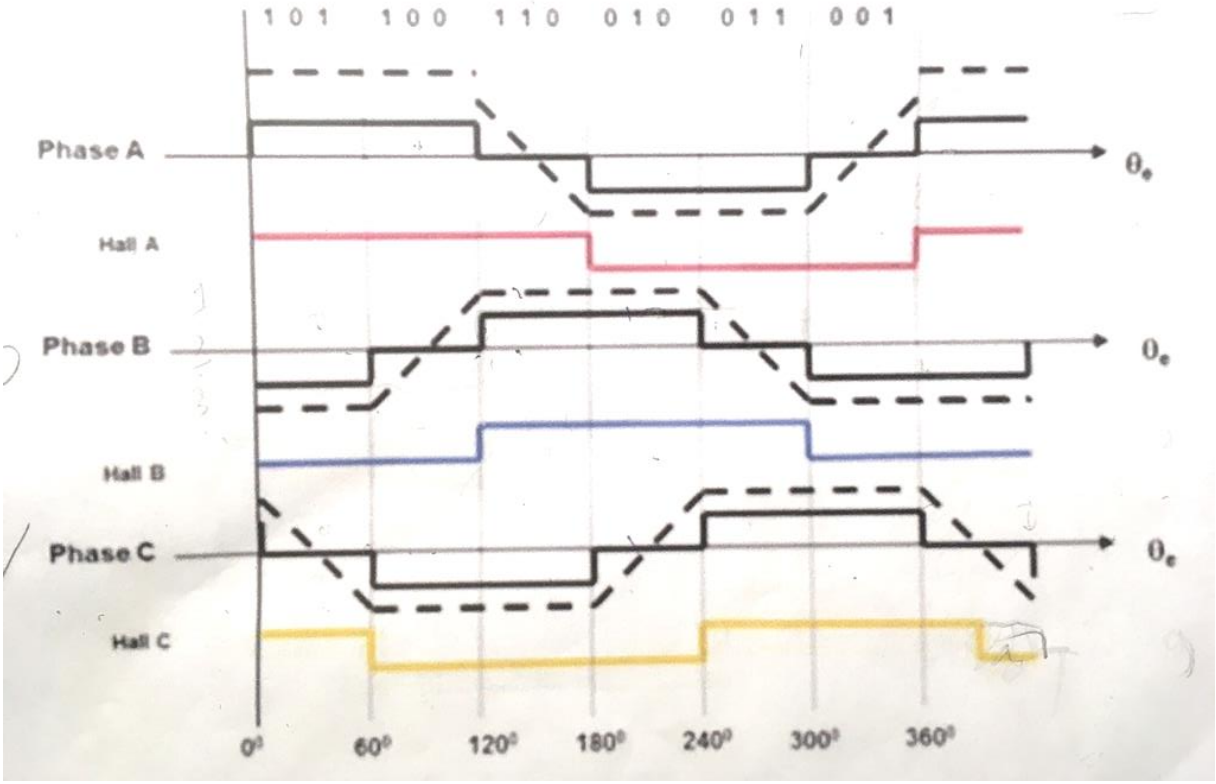
de MOSFET en conduction. Un circuit logique AND 74HC08 distribue le signal même signal PWM sur les trois transistors MOSFET du bas.

Les MOSFET ont remplacé des IGBT qui chauffaient trop en saturation (V_{CEsat} 1,3V x Ic 10A = 13W) alors que les MOSFET dissipent dix fois moins (R_{on} 0,013Ω x Is x Is 10A= 1.3W)

Des MOSFET 75NF75 qui supportent 75A ont remplacé les IRF540 trop limités en courant.

L'estimation de la fréquence des phases moteurs est pour 25km/h = 11,6 tours/s = 11,6*15 = 174 Hz

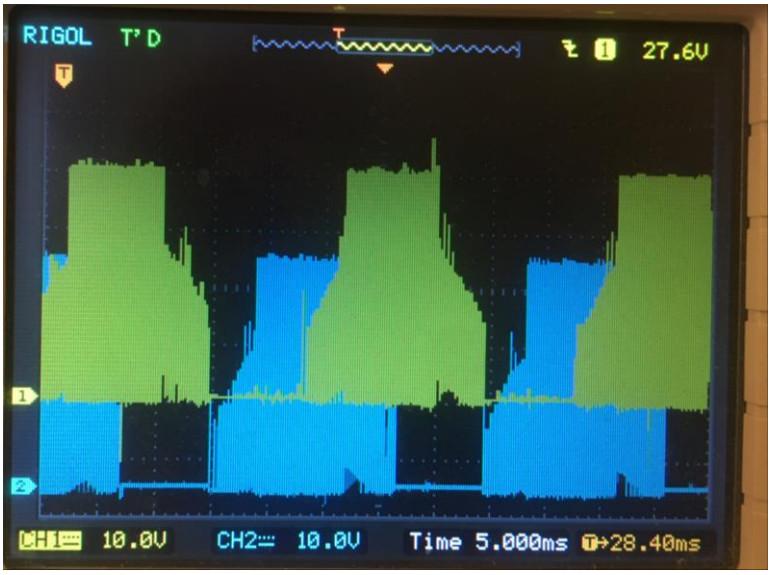
Le séquencement des phases du moteur suit la logique suivante :



En haut les états des capteurs Hall, plus bas
les phases A,B,C. (ou jaune, bleu, verte)

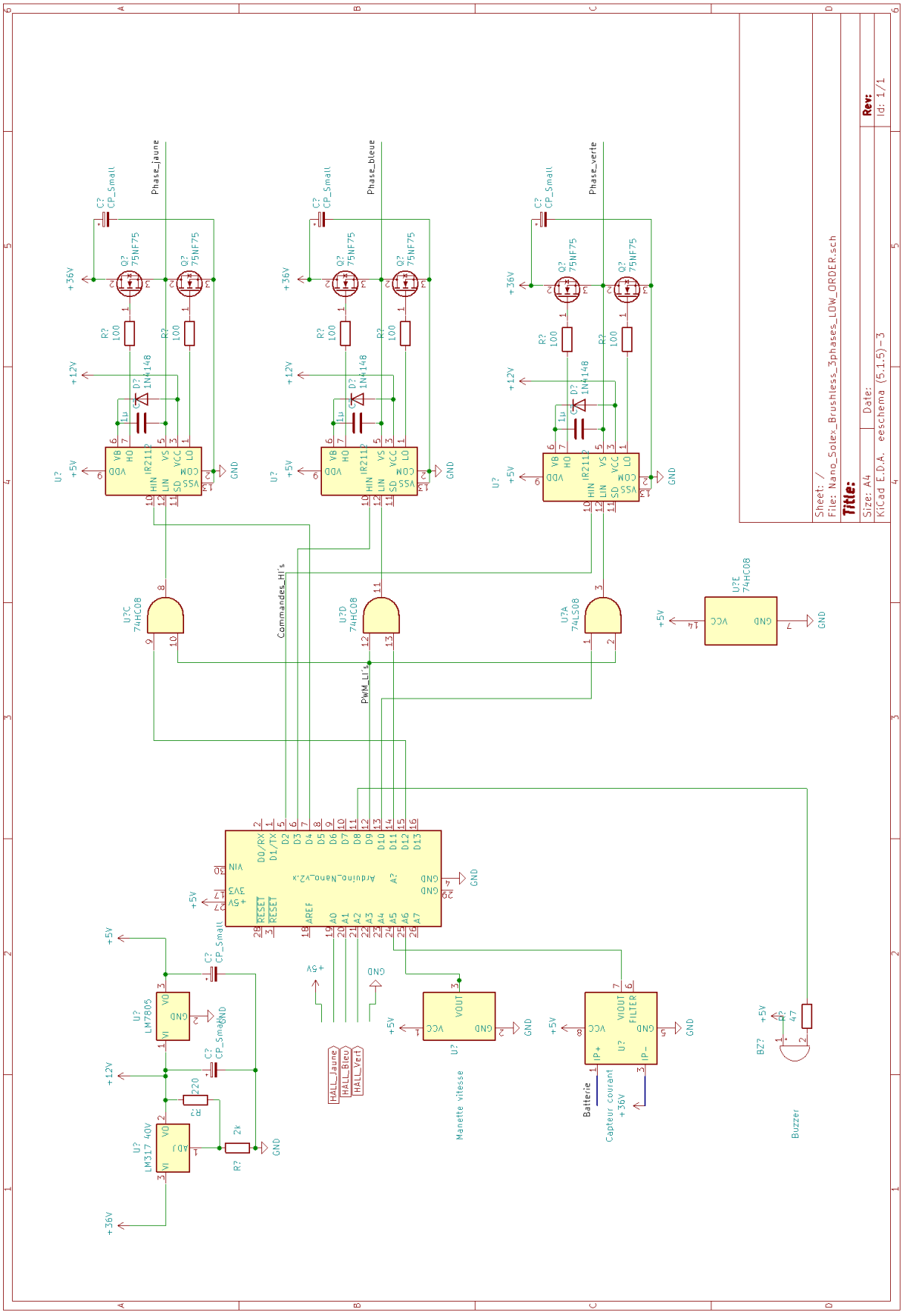
Les mesures sur le contrôleur

Les formes d’ondes observées:

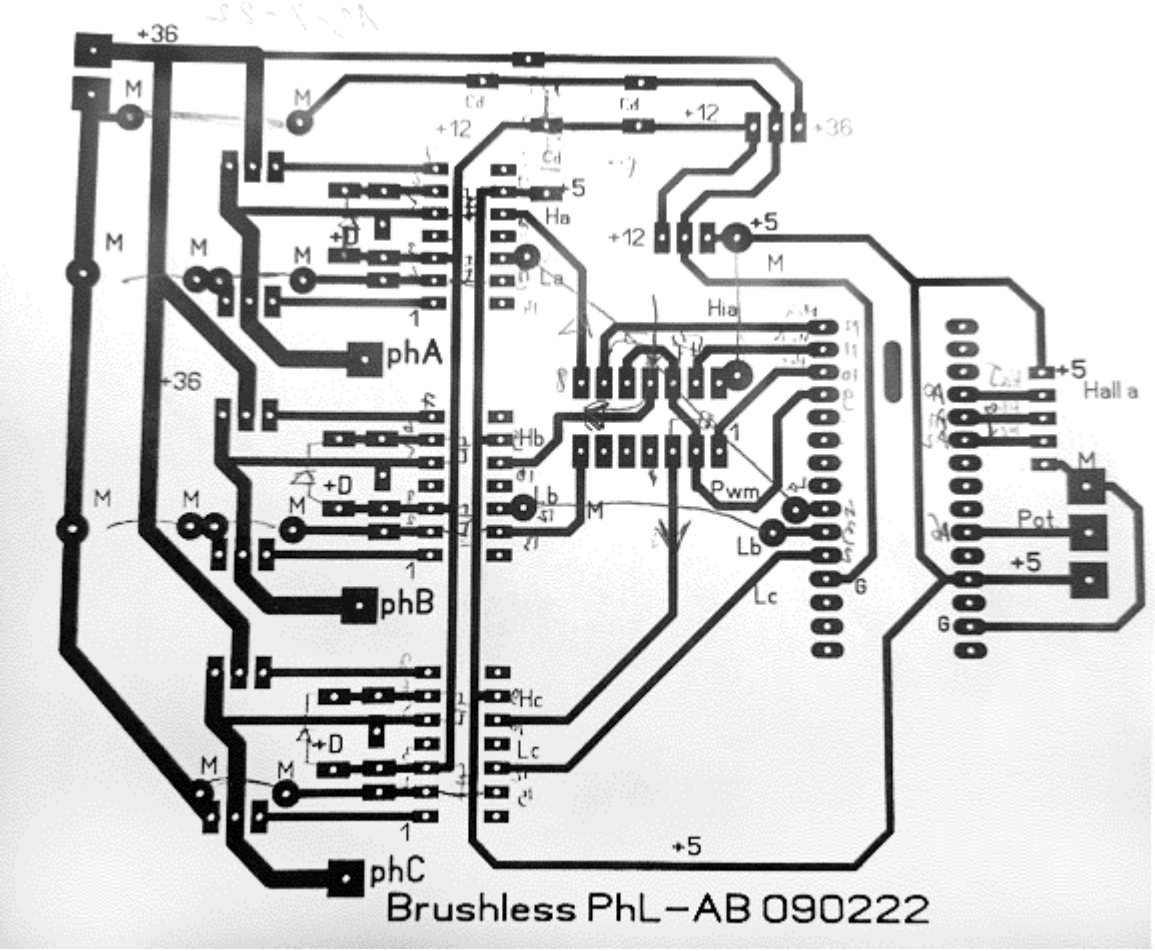


Les signaux PWM des phases jaunes et bleux (déphasage 120°)

Le signal reçu du capteur Hall et la phase correspondante (ratio PWM =255)



Le schéma électrique du contrôleur



Le circuit imprimé du prototype coté soudure (avant modifications)

Le montage sur le Solex

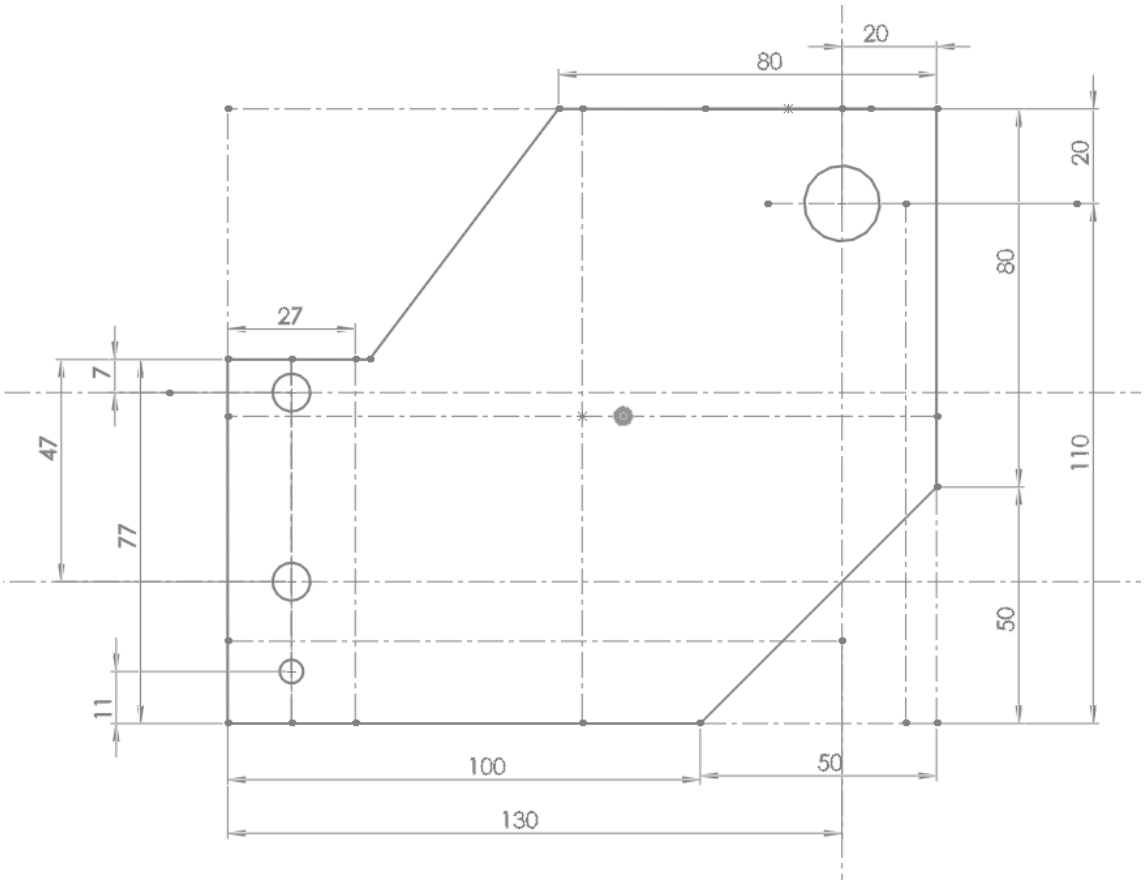
La dépose du moteur thermique

Cinq vis retiennent le moteur, le réservoir et le pot d'échappement

Le montage du moteur électrique

L'utilisation des deux vis de fixation des fourches avant du Solex

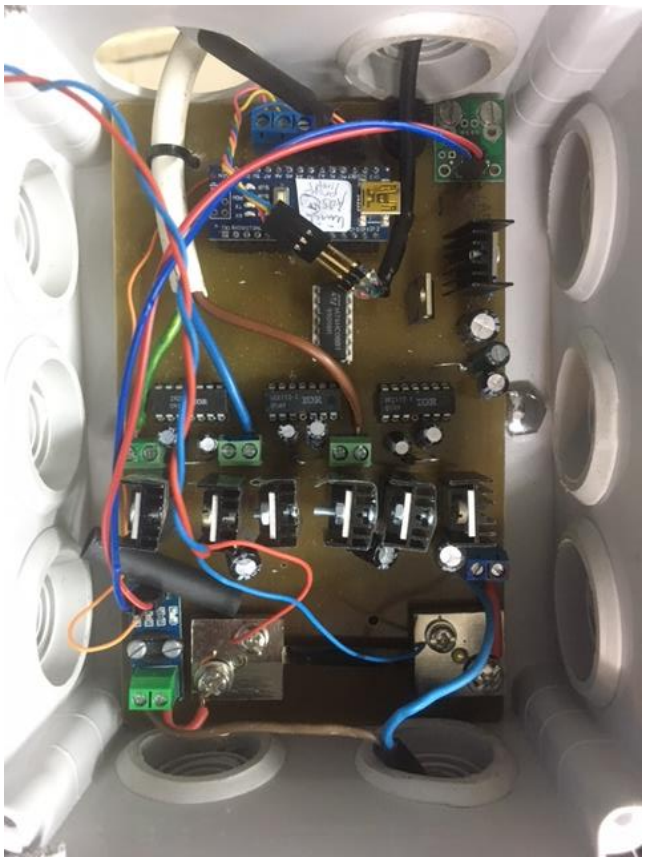
La fixation du moteur pour essai sur une plaque en bois puis sur une plaque d'aluminium d'épaisseur 5mm



La découpe et le perçage (16mm, 8mm et 5mm) de la plaque support moteur

La manette d'accélérateur installée à la place de la manette de décompression

Le boitier étanche et son ampèremetre 30A pour surveiller la puissance



L’électronique de contrôle et les connexions de câbles

La batterie au lithium 36V 10Ah sur le porte bagage.

Les performances estimées

Vitesse maximum: 25 km/h

Autonomie : 1 heure à vitesse maximum

Fin du document

La bicyclette qui roule et monte sans bruit

Annexe

Code Arduino

```
// Auteurs Philippe Loutrel / Alain Bertout
//Brushless__Nano_IR2112_130222
//Chaque IR2112 pilote une paire de FET.Bonne saturation, Vg à 12V
//Un cycle= Pwm seulement sur une phase, masse/flottement sur les 2 autres
//Une seule sortie D9 en PWM reparti par porte AND 74HC08
//3 phases et 3 capteurs Hall: Jaune, Bleu, Vert fils de 3 couleurs(ou A,B,C)
//Hypothèse: alimentation des FET sous 36V, IR2112 pin sous  12V
//Les FET ont remplacé les IGBT qui chauffent trop en saturation sous 1.3V
Ω//PWM  sur sortie D9 distribué par 3 portes AND (74HC08) aux 3 IR2112
//Ratio (rapport cyclique)Rc de PWM variables par gachette commandée au pouce ( origine trottinette electrique )
//La phase J peut être connectée en au +36V HiJ1/LoJ0 au +, ou au GND HiJ0/LoJ1  au -, ou restée flottante HiJ0/LoJ0 (non alimentée)
//Pour chaque phase il y a donc un couple de FET  : (HiJ,LoJ), (HiB,LoB) et (HiC,LoC)
//Creneau unitaire: 60° electrique(une unité )  soit 4° angle de rotation
//Une connexion au + ou - dure 120°
//360°electrique = 24° rotation = 6u.
//1 tour de roue = 15*24°
//Diamétre de roue = 19cm *pi = 0.6m.
// Pour 25km/h = 11.6 tours/s = 11.6*15 = 174 Hz
//Quand un capteur Hall change d'état, une  phase connectée devient flottante(deconnection)
//R phase = 0.25 Ohm et L phase =0.3mH
//Toujours deux phases en série donc 0.5 Ohm et 0.7mH


//Sortie Pwm, affectable à HiJ,ou HiB,ou HiV via porte AND HC7408
#define HallJ A0   //Capteur Hall  de la phase Jaune, sous 5V, et R pullup du Nano
#define HallB A1    //Capteur Hall  de la phase Bleue
#define HallV A2    //Capteur Hall  de la phase Verte
#define ShutDown A3  // Chutdown IR2112 ( actif HIGH)
#define Iport2 A4 // Capteur courant tore
#define Iport A5    // Analog Capteur courant ACS714
#define Pot A6      // Analog Potar en lecture pour varier Rc
#define LoJ 12     // Vers grille FET Lo de la phase jaune,
#define  LedPin 13
#define  BuzzPin 8
bool stat = HIGH;
#define LoB 11     //Phase bleue
#define LoV 10     //Phase verte
#define HiJ 4      //vers grille FET Hi de la phase Jaune
#define HiB 3       //Phase bleue
#define HiV 2       //Phase verte
//FET Hi   Grille commandée via une porte AND et IR2112
#define LoJ1 digitalWrite(LoJ,1) // Controle la grille du FET LoJ: 1 phase Jaune connectée au Pwm
#define LoJ0 digitalWrite(LoJ,0) //grille = 0, la phase Jaune est flottante ou à la masse
#define LoB1 digitalWrite(LoB,1)  //Avec LoJ0 && LoJ0, la phase J FLOTTE
#define LoB0 digitalWrite(LoB,0)
#define LoV1 digitalWrite(LoV,1)
#define LoV0 digitalWrite(LoV,0)
//FET Lo   Grille commandée via IR2112
#define HiJ1 digitalWrite(HiJ,1) // Controle la grille du FET HiJ:1 phase Jaune à la masse
#define HiJ0 digitalWrite(HiJ,0) // grille = 0, la phase Jaune est flottante ou au Pwm
#define HiB1 digitalWrite(HiB,1)
#define HiB0 digitalWrite(HiB,0)
#define HiV1 digitalWrite(HiV,1)
#define HiV0 digitalWrite(HiV,0)

int Rc = 0;       //
int jbv = 0;       //6 états possibles du système 001 à 110, selon les capteurs Hall
int prev_jbv; // état précedent de jbv
int Courant; //Courant mA
int Ic;
int CourantACS; // capteur Hall ACS712
int courant4, courant3, courant2, courant1, courant ; // pour moyenne courant
```

```
long int  k;
int delta_I = 0;
float PWR;

void setup() {
  Serial.begin(115200);
  pinMode(HallJ, INPUT_PULLUP); // pour entrée capteur Hall jaune
  pinMode(HallB, INPUT_PULLUP); // pour entrée capteur Hall bleu
  pinMode(HallV, INPUT_PULLUP); // pour entrée capteur Hall vert
  pinMode(HiJ, OUTPUT); pinMode(LoJ, OUTPUT); //Sorties vers les grilles des 2 FET d'une phase
  pinMode(HiB, OUTPUT); pinMode(LoB, OUTPUT);
  pinMode(HiV, OUTPUT); pinMode(LoV, OUTPUT);
  pinMode(ShutDown, OUTPUT); // Shutdown IR2112
  pinMode(LedPin, OUTPUT); // LedPin pin 13
  pinMode(BuzzPin, OUTPUT); /// buzzer pin 8 courant max
  TCCR1B = TCCR1B & B11111000 | B00000010 ; // pour une fréquence PWM de 3921,16 Hz sur pin 9
  pinMode(9, OUTPUT); /// PMW natif avec prédiviseur = 8 sur timer 1
  // lire https://passionelectronique.fr/pwm-arduino/#broches-de-sortie-pwm-output-pins-arduino
  // mise à zero des grilles de MOSFET
  LoJ0; // reset PMW J
  LoB0; // reset PMW B
  LoV0; // reset PMW V
  HiJ0; // reset  J
  HiB0; // reset  B
  HiV0; // reset  V
} // fin setup


///**********************LOOP**************************************************
void loop() {

  //////reglage Rc en fonction du capteur de courant ACS712 et de la Consigne Ic ( accelerateur à pouce )
  Ic = 3000;  // courant max
  Courant = LireCourant();
  Rc = LireConsigne();
  Moteur (Rc); // passage de Rc en parametre

  while (Courant > Ic) // tant que le courant est supérieur  à la consigne
  {
    digitalWrite(LedPin, HIGH);
    digitalWrite(BuzzPin, LOW);
    Courant = LireCourant();
    delta_I = (Courant - Ic) / 100;
    // réduction progressive du rapport cyclique Rc du PWM ------------------------------------------
    Rc = Rc - delta_I;
    if (Rc <= 0) {
      Rc = 0;
    }
    Moteur (Rc); // passage de Rc en parametre
  } // fin du while
  digitalWrite(LedPin, LOW);
  digitalWrite(BuzzPin, HIGH);

} // fin du loop()

////////////////************************************** Les fonctions moteurs ************************
/*
  Il faut éviter toute erreur dans la séquence des codes JBV venant des capteurs Hall ( pas de supposition de logique d'enchainement  d'état par défaut )
  On confirme donc tous les états des transistors MOS High et Low pour chacune des 6 phases:
*/

void  HallB_up() //221
{
  LoB0; HiB1;  //36V B
  LoJ0; HiJ0; //Flotte J
  HiV0; LoV1;   //PWM V
}

void  HallB_down() //211
```

```
{
  LoV0; HiV1; //36V V
  LoJ0; HiJ0; //Flotte J
  LoB1; HiB0; //PWM B
}

void  HallV_up() //221
{
  LoV0; HiV1;  //36V V
  HiB0; LoB0; //Flotte B
  LoJ1; HiJ0; //PWM J
}

void  HallV_down() //121
{
  LoJ0; HiJ1; //36V J
  LoB0; HiB0; //Flotte B
  HiV0; LoV1;  //PWM V
}

void  HallJ_up() //122
{
  LoJ0; HiJ1;  //36v J
  LoV0; HiV0; //Flotte V
  HiB0; LoB1;  //PWM B
}

void  HallJ_down() // 112
{
  LoB0; HiB1;  //36V B
  LoV0; HiV0; //Flotte V
  HiJ0; LoJ1;  //PWM J
}

void  Shut_down()
{
  LoB0; HiB0;  //Flotte B
  LoV0; HiV0; //Flotte V
  HiJ0; LoJ0;  //Flotte J
  digitalWrite (ShutDown, HIGH); //option SD IR2112
}

//**********************************Les Capteurs ****************************************
int LireCourant ()
{
  CourantACS = analogRead(Iport2);   //Serial.print( Courant);// mesure Courant
  if (CourantACS >= 629) {
    CourantACS = 629;  // 509 corresp 0, 629 correspond à 10A
  }
  Courant = map(CourantACS, 512, 937, 0, 30000);  // milliA Capteur ACS712  //  Serial.print ( "ACS= ");
  return (Courant);
}

int LireConsigne()
{
  //////lecture consigne de vitrddr  Gachette d'accélération à pouce
  int Rcc = analogRead(Pot) ; // Consigne vitesse ( rapport cyclique PWM)
  Rcc = map(Rcc, 189, 860, 0, 255); // accelerateur à capteur Hall consigne courant plage 0-10000mA);
  if (Rcc >= 255) {
    Rcc = 255;    //Butée haute
  }
  if (Rcc < 0) {
    Rcc = 00;       //Butée basse
  }
  return (Rcc);
}
```

```
//********************************** commande du moteur ***************************************
void Moteur(int Ratio) // passage du rapport cyclique PWM : Rc
{

  //LECTURE DES CAPTEURS HALL MOTEUR BRUSHLESS
  //  L'état du système jbv est calculé en fonction des capteurs Hall
  jbv = (1 + digitalRead( HallJ)) * 100 + (1 + digitalRead (HallB)) * 10 + (1 + digitalRead (HallV));

  //SEQUENCEUR TRIPHASE
  analogWrite(9, Ratio); // PWM natif
  switch (jbv)
  {
    case (212): HallJ_up(); break;  //(J reste bas), B flotte, V pwm
    case (211): HallV_down(); break;//J flotte, B masse, V (reste pwm)
    case (221): HallB_up(); break;  //J au pwm, B (reste bas), V flotte
    case (121): HallJ_down(); break;//J (reste pwm), B flotte, V masse
    case (122): HallV_up(); break;  //J flotte, B au pwm, V (reste bas)
    case (112): HallB_down(); break;//J masse,B (reste pwm), V flotte
  }
} // fin du void moteur()
```