



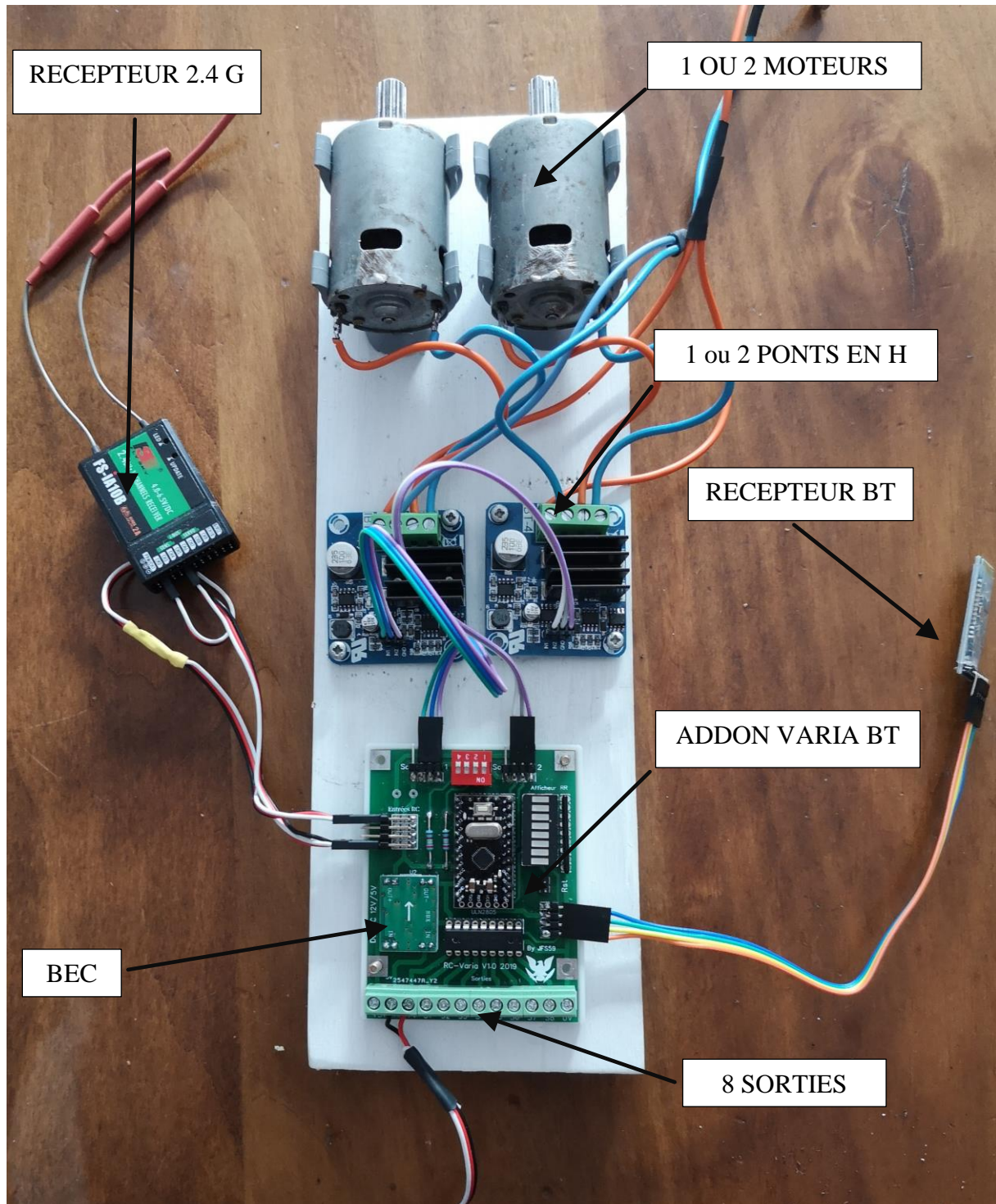
# DOUBLE VARIATEUR DE VITESSE POUR BATEAUX



Disponible sur <https://github.com/jfs59/Variateur-BT>

**Merci de bien lire le descriptif.**

JFS59 @ 2019-2020



## Réalisation d'un variateur double pour bateau et d'un ensemble de 8 voies supplémentaires.

### Préambule :

Librement inspiré de : [http://bateaux.trucs.free.fr/huit\\_sorties\\_auto\\_v3.html](http://bateaux.trucs.free.fr/huit_sorties_auto_v3.html) je n'ai gardé pratiquement que l'idée, l'ensemble du code, les schémas ont été réécrits et optimisés tant au niveau du décodage de la mémorisation et des options en fait il reste très peu sinon rien du travail de M Claverie à part l'idée (Il suffit de regarder les extraits de code pour s'en convaincre) qu'il en soit remercié. Certains commentaires sont en Anglais ! Je le déplore mais c'est malheureusement plus concis et souvent plus simple que le Français. La première version était une simple extension sans variateur mais la demande pressante des membres du club on fait que... Cette version bien que non décrite existe toujours et elle est finalisée. (Sauf pour la partie mp3)

### Matériel :

Arduino pro mini, afficheur barre de led, réseau de résistance, ULN 2005, Bornes de sorties pour circuit imprimé, convertisseur DC/DC 12V → 5V, IBT-4 (pont en H), Bluetooth JDY-30, Mp3 à définir.

### Principe :

L'Arduino décode les signaux RC PPM (4 canaux) et génère les pwm nécessaires à la commande de deux IBT-4 (pont en H) à partir des canaux 1 et 2.

Si fonctionnement sur un seul canal des deux moteurs :

Le canal 2 est affecté au gouvernail et permet suivant le mode de fonctionnement de fonctionner en moteur différentiel.

Le canal 3 est décodé pour activer le canal 4 en sortie ou en demande de bruiteur (module mp3 facultatif)

Le canal 4 est décodé pour commander 8 sorties ULN 2005 selon le principe d'extension de voie. (8 voies sur un canal). En fonction du canal 3 il permet de commander 8 bruits différents, (corne, cloche, diesel)

L'ensemble est configurable par liaison Bluetooth à l'aide d'un programme spécifique développé pour Android. Un mode simulateur est disponible.

Sont configurables : (En plus du mode de fonctionnement)

Limite haute télécommande Canal 1 : 2000 par défaut.

Limite basse télécommande Canal 1 : 1000 par défaut.

Neutre Canal 1 : 1500 par défaut.

Limite haute télécommande Canal 2 : 2000 par défaut.

Limite basse télécommande Canal 1 : 1000 par défaut.

Neutre Canal 2 : 1500 par défaut.

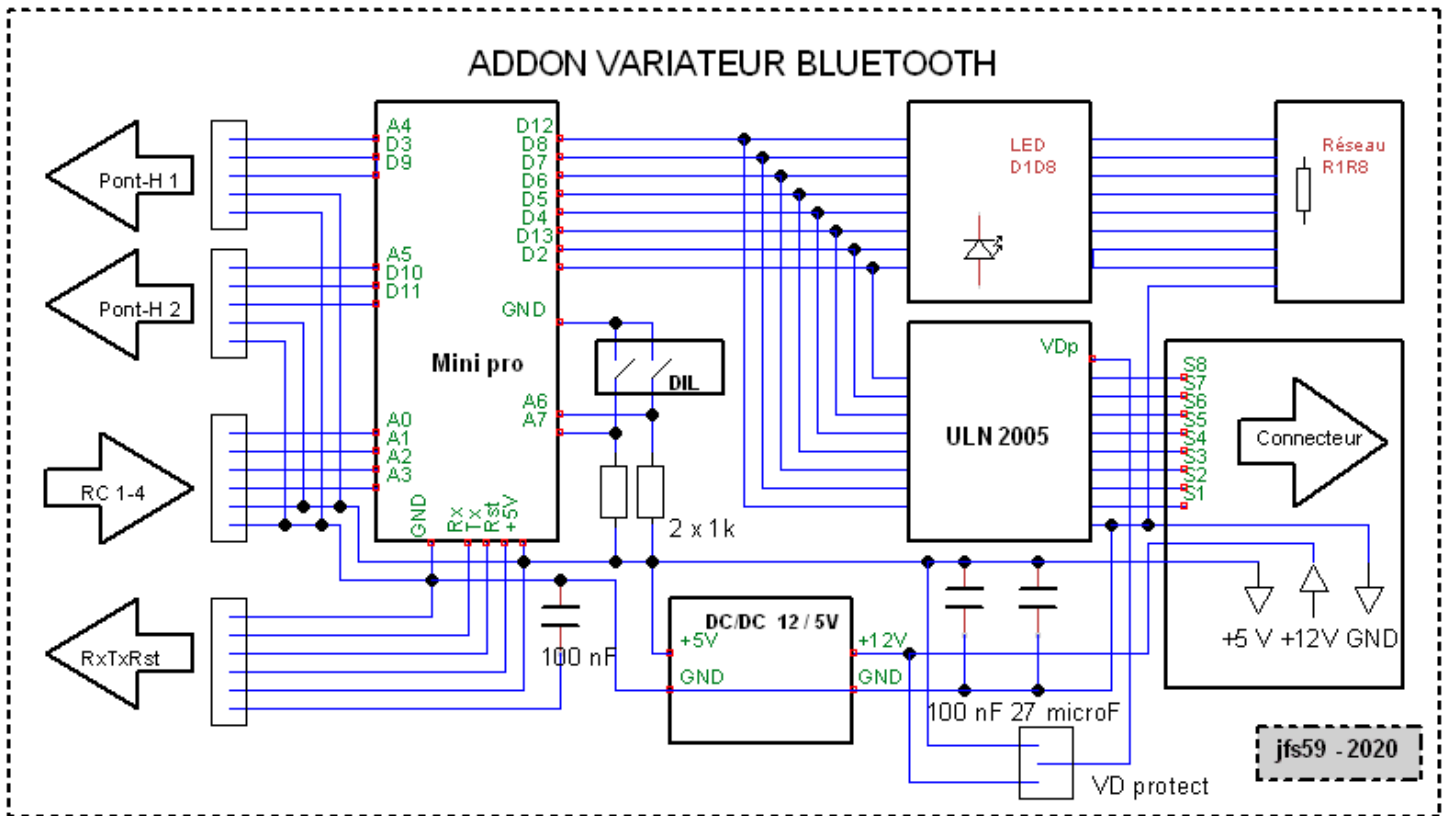
Largeur zone neutre : +/- 50 par défaut.

Pourcentage AV Pourcentage AR Canal 1 : 100% par défaut.

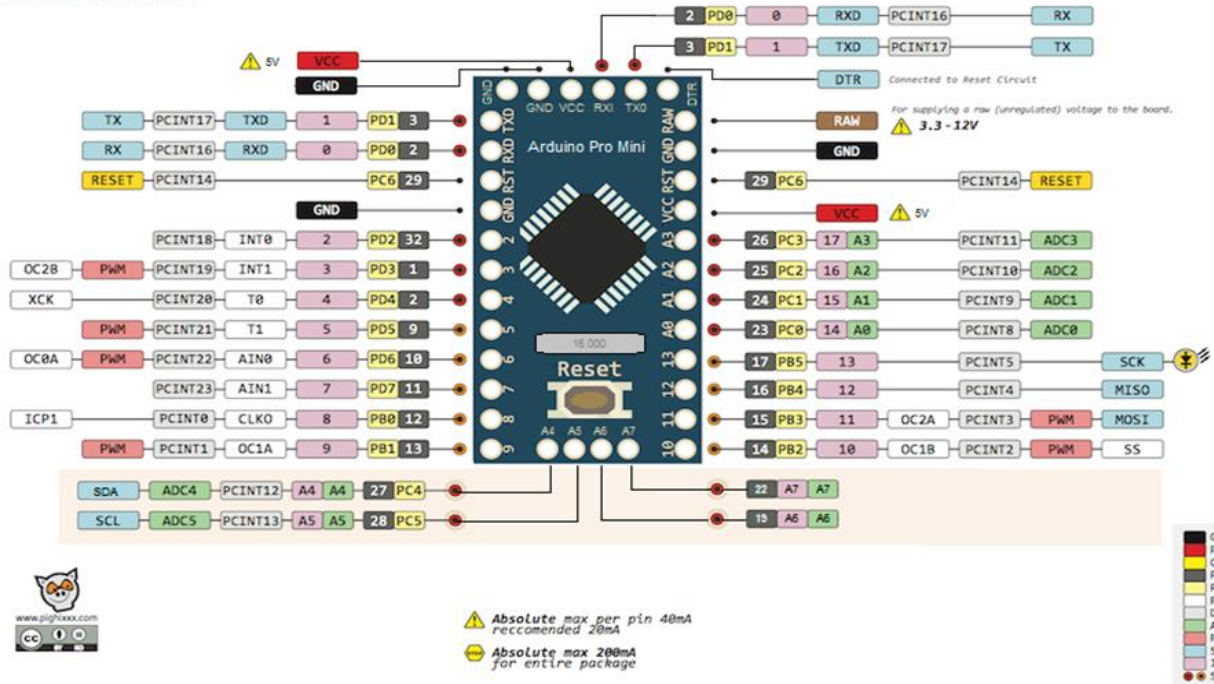
Pourcentage AV Pourcentage AR Canal 2 : 100% par défaut.

Mode de fonctionnement des 8 sorties : Monostables Bistables : 4 monostables, 4 bistables par défaut.

## SCHEMA DE PRINCIPE



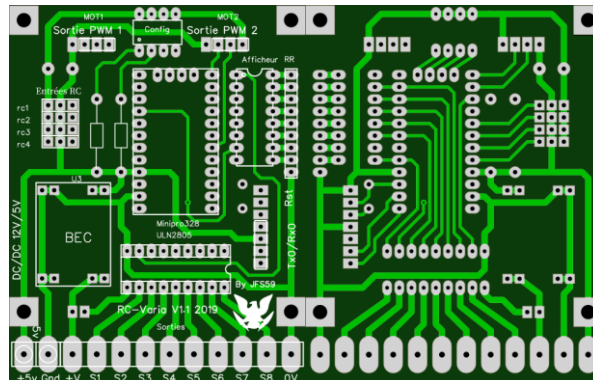
## ARDUINO MINI PRO 5V / 16 MHz





## CIRCUIT IMPRIME

Circuit double face, trous métallisés, vernis épargne, et sérigraphie. Dessiné et routé par mes soins. Réalisé par une entreprise. (Le résultat donne un CI professionnel) Le routage et le design sont donnés ci-dessous mais les fichiers Gerber sont et resteront ma propriété. Le circuit imprimé V3 est disponible pour la somme de 5 € couvrant les frais de réalisation et transport. ( Me contacter au club de Raismes. )



Remarque : l'utilisation de A6 et A7 en numérique implique des résistances de tirage au +. A6 et A7 étant purement analogique il faut utiliser une astuce de programmation.

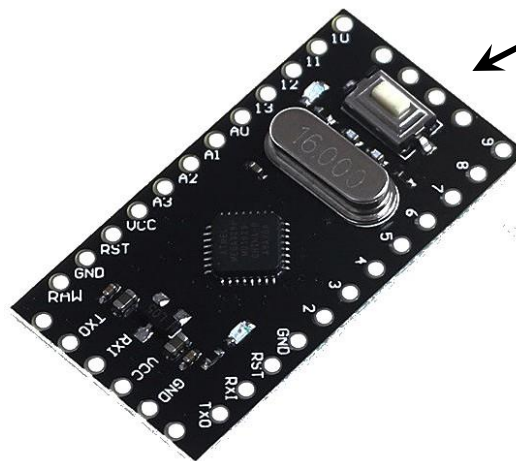
Le bootloader de la carte devra être changé pour empêcher le clignotement au reset de la carte ! sinon il faut faire attention au rôle de la sortie S7 qui va s'allumer au reset !!

Dans la version 3 le circuit imprimé a changé car la programmation Bluetooth a pris le dessus sur la configuration par DIL (cf. schéma de principe) des sorties supplémentaires pour les ponts en H sont disponibles. Elles permettront d'utiliser des ponts en H autre que l'IBT4.

L'entrée commune de protection par diodes de l'ULN peut être reliée soit au 12 V soit au 5 V (version 3) soit restée non utilisée (cas de tensions différentes sur les sorties)

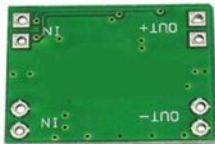
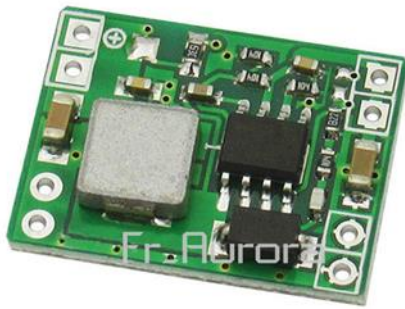
**Bien vérifier à la commande car de nombreuses variantes incompatibles !!**

**ATTENTION : il faut un Pro Mini Atmega328P 5V/16M avec le même brochage du CI de [ A4 à A7 ]**



## Composants et sous-ensembles.

### Convertisseur DC/DC 12 V/ 5 V

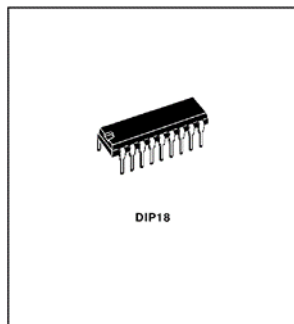


Utilisé pour alimenter l'Arduino et les diodes de visualisation. Une sortie 5 V est prévue pour alimenter un montage externe.

Sur la première version montage du convertisseur composant en dessous !

Version 3 le convertisseur est positionné composants au-dessus.

- EIGHT DARLINGTONS WITH COMMON EMITTERS
- OUTPUT CURRENT TO 500 mA
- OUTPUT VOLTAGE TO 50 V
- INTEGRAL SUPPRESSION DIODES
- VERSIONS FOR ALL POPULAR LOGIC FAMILIES
- OUTPUT CAN BE PARALLELED
- INPUTS PINNED OPPOSITE OUTPUTS TO SIMPLIFY BOARD LAYOUT



DIP18

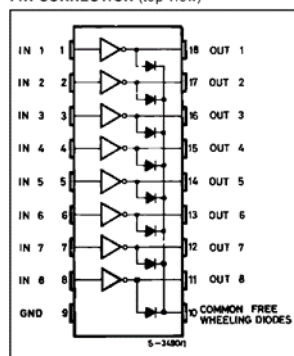
#### DESCRIPTION

The ULN2801A-ULN2805A each contain eight darlington transistors with common emitters and integral suppression diodes for inductive loads. Each darlington features a peak load current rating of 600mA (500mA continuous) and can withstand at least 50V in the off state. Outputs may be paralleled for higher current capability.

Five versions are available to simplify interfacing to standard logic families: the ULN2801A is designed for general purpose applications with a current limit resistor; the ULN2802A has a 10.5kΩ input resistor and zener for 14-25V PMOS; the ULN2803A has a 2.7kΩ input resistor for 5V TTL and CMOS; the ULN2804A has a 10.5kΩ input resistor for 6-15V CMOS and the ULN2805A is designed to sink a minimum of 350mA for standard and Schottky TTL where higher output current is required.

All types are supplied in a 18-lead plastic DIP with a copper lead from and feature the convenient input-opposite-output pinout to simplify board layout.

#### PIN CONNECTION (top view)

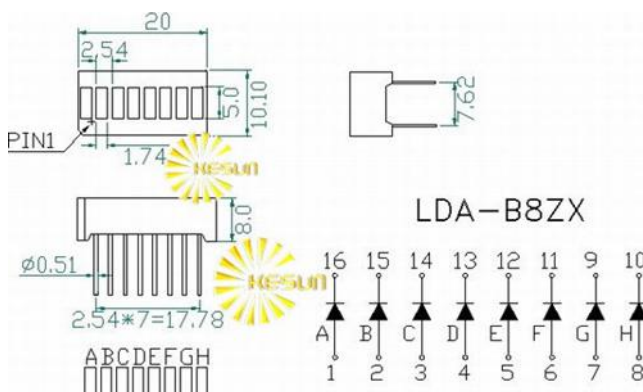


### ULN 2005 interface de puissance

Permet de relayer les sorties Arduino afin de disposer d'un courant de sortie plus important et d'une tension différente, peuvent se mettre en // pour augmenter le courant de sortie il suffit de les souder patte pour patte l'un au-dessus de l'autre,

Diode anti retour de protection communes a toutes les sorties.

Les sorties OUT sont mises à la masse quand un niveau haut (1) est présent à l'entrée correspondante.



### Bargraphe led

Visualise l'état des canaux et donc des sorties de puissance. Les cathodes sont reliées à la masse à travers un réseau des 8 résistances dont le point commun est à la masse.

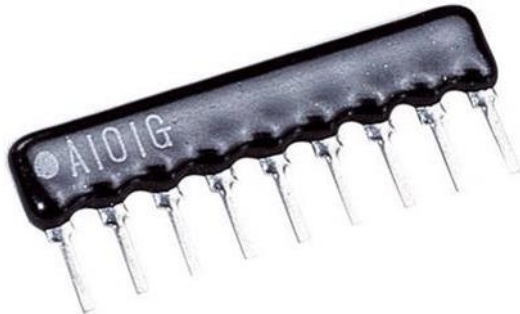
## Interrupteur DIL



Permettent de choisir le mode de fonctionnement de 0 à 15. Depuis la version Bluetooth le mode est configurable par le programme Android.

Le DIP ne fait plus que 2 rangées sur la version 3.

## Réseau de 8 résistances



Choisir la valeur entre 470 Ohm et 1 K Ohm,

## Borniers a vis clipsables,

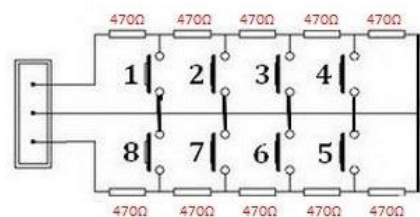


Par 2 ou 3 il en faut 12 : 2 X6 ou 3 X4

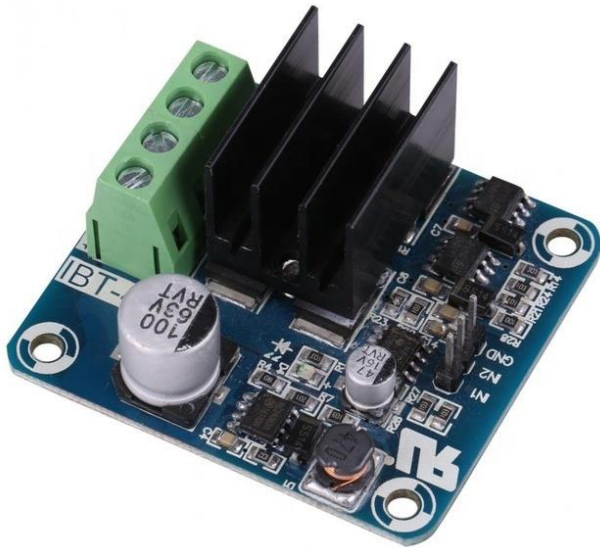


Un fichier imprimante 3D est disponible sur le Github

Schéma de câblage du clavier



## IBT4 Pont en H



Caractéristiques :

- Module driver Pont H à courant élevé (50A) MOSFET
- Isolation galvanique du signal PWM.
- Rotation avant et arrière du moteur, deux entrées PWM au maximum 200kHz
- Alimentation 3.3V à 15V.

Le module possède sa propre alimentation donc seulement 3 fils à brancher la masse et les deux entrées PWM.

## Double Pont en H



Caractéristiques :

- Module driver Pont H (10 A ou 15 A) MOSFET

Courant nominal : 10A

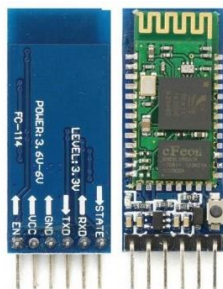
Courant de crête : 30A

Moteur avant : DIR = 1 PWM = PWM

Inversion du moteur : DIR = 0 PWM = PWM

Frein de stationnement : DIR = X PWM = 0 (X est un état arbitraire)

(Pas compatible encore en développement)



## Module Bluetooth HC 05 / JDY-30

Il permet la configuration du variateur (mode de fonctionnement) et des différentes sorties (monostable, bistable) il permet aussi un mode simulation des sorties et des voies du variateur.

# Chronologie de câblage :

## Câblage de la version Bluetooth.

Souder les deux résistances cms 1K et vérifier en testant sur le circuit.

Souder la platine alimentation BEC.

Souder une ligne de bornier (possibilité de souder : +V, gnd, +5v uniquement)

Brancher une tension >6 V sur Gnd, +V et vérifier la présence du + 5 V.

Souder le connecteur Rx/Tx Bluetooth et le condensateur c3 de 100 nf. (Pour permettre le flash de la carte Arduino il est aussi possible de flasher à la volée par le connecteur « ftdi »)

Souder les pattes extrêmes du réseau et vérifier le sens et les valeurs. (Souder le reste des pattes)

Souder le support d'afficheur. Placer l'afficheur et vérifier chaque segment en injectant +5 v sur les pastilles correspondantes au niveau de la carte Arduino.

Souder la plaque Arduino MiniPro. (Eventuellement le bootloader noled aura été chargé mais il peut l'être après)

Charger un sketch (programme) dans l'Arduino.

Mettre sous tension entre Gnd et +V (tension > 6V). Le programme démarre est on doit observer à chaque reset un défilement « bargraphe » sur l'afficheur.

Souder le support ULN.

Placer un ULN et vérifier le bon fonctionnement de chaque sortie (S1 a S8). (Terminer les soudures borniers si nécessaire)

Souder l'interrupteur DIL éventuellement sur support.

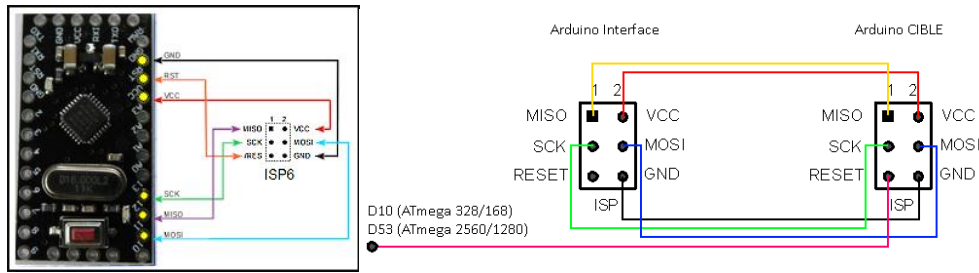
Souder les connecteurs de sortie PWM.

(Procédure incomplete)



# Compilation et Flash du Bootloader

Repérage des connecteurs de la mini pro :



Sur le circuit imprimé vous pouvez utiliser les broches support de l'afficheur.

Se référer à la procédure de flash soit avec un ISP soit avec un Arduino câblé en ISP et le programme (Arduino as ISP).

Le bootloader zéros LED est disponible sur le GitHub.

**Il faut évidemment placer le bootloader noled dans le dossier Arduino et éditer le Board.txt !**

[https://github.com/jfs59/Variateur-BT/blob/master/mod.ATmegaBOOT\\_168\\_atmega328.hex](https://github.com/jfs59/Variateur-BT/blob/master/mod.ATmegaBOOT_168_atmega328.hex)

Ce bootloader no LED a été compilé par mes soins et fonctionne avec la mini pro sans problème. Il permet de supprimer le clignotement lors du boot du variateur. (Il n'est pas indispensable si l'activation de la sortie au démarrage ne pose pas de problème sur la maquette)

**Compiler un bootloader :** (Pour rappel ou besoin différent)

charger et installer GnuWin32

copier : ATmegaBOOT\_168.c et Makefile

dans le dossier bin de gnuwin32

(pour aller dans le dossier en ligne de commande : `cd \"program files (x86)\GnuWin32\bin\"` )

dans le makefile : modifier (modification du nombre de clignotement au boot)

**atmega328: CFLAGS += '-DMAX\_TIME\_COUNT=F\_CPU>>4' '-DNUM\_LED\_FLASHES=1' -DBAUD\_RATE=57600**

en mettant zéro à la place du 1

**atmega328: CFLAGS += '-DMAX\_TIME\_COUNT=F\_CPU>>4' '-DNUM\_LED\_FLASHES=0' -DBAUD\_RATE=57600**

Lancer un : **`./make atmega328`** (ne pas oublier `./`)

```
PS C:\program files (x86)\GnuWin32\bin> ./make atmega328
avr-gcc -g -Wall -Os -mmcu=atmega328p -DF_CPU=16000000L '-DMAX_TIME_COUNT=F_CPU>>4' '-DNUM_LED_FLASHES=0' -DBAUD_RATE=57600 -c
-o ATmegaBOOT_168.o ATmegaBOOT_168.c
avr-gcc -g -Wall -Os -mmcu=atmega328p -DF_CPU=16000000L '-DMAX_TIME_COUNT=F_CPU>>4' '-DNUM_LED_FLASHES=0' -DBAUD_RATE=57600 -
Wl,--section-start=.text=0x7800 -o ATmegaBOOT_168_atmega328.elf ATmegaBOOT_168.o
avr-objcopy -j .text -j .data -O ihex ATmegaBOOT_168_atmega328.elf ATmegaBOOT_168_atmega328.hex
rm ATmegaBOOT_168_atmega328.elf ATmegaBOOT_168.o
PS C:\program files (x86)\GnuWin32\bin>
```

Renommer le fichier obtenu en **mod.ATmegaBOOT\_168\_atmega328.hex**

puis le transférer dans le dossier C:\Program Files (x86)\Arduino\hardware\arduino\avr\bootloaders\atmega

Editer le boards.txt et changer :

**pro.menu.cpu.16MHzatmega328.bootloader.file=atmega/mod.ATmegaBOOT\_168\_atmega328.hex**

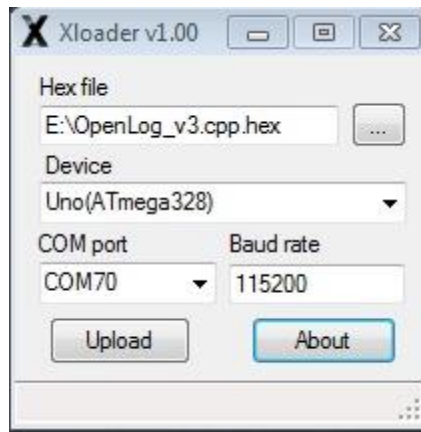
dans la section pro mini

enfin lancer l'ide arduino, et avec un nano et le programme arduinoisp faire un flash arduino as isp du bootloader

lancer 2 fois si le premier ne passe pas ! Puis recharger le sketch.

Pour flasher le .hex vous pouvez utiliser Xloader en USB ou une autre méthode que vous connaissez.

### Xloader



A rechercher sur internet ou directement sur le GitHub <https://github.com/jfs59/Variateur-BT/blob/master/Outils/XLoader.zip>

Choisir le port com et le fichier .hex

La procédure fonctionne je l'ai essayée.

## **Gestion des paramètres en Bluetooth sur ANDROID**

### Varia config 1.0:

Varia Config 1.0 (et les futures évolutions) permet de gérer les paramètres du variateur à l'aide d'un téléphone ou d'une tablette ANDROID. (Il n'y aura pas d'autres développement que sur ANDROID)

Développé par mes soins à partir de <http://ai2.appinventor.mit.edu/>

Le programme a beaucoup évolué (et évoluera encore) en fonction des options ajoutées et des besoins.

Il est disponible ici : <https://github.com/jfs59/Variateur-BT/blob/master/Variateur.apk>


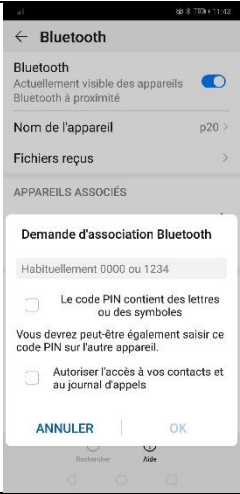
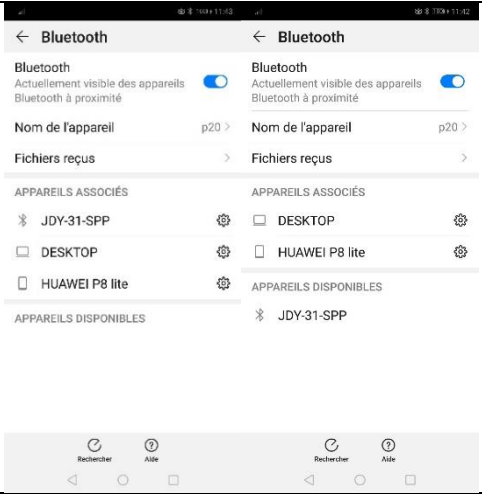
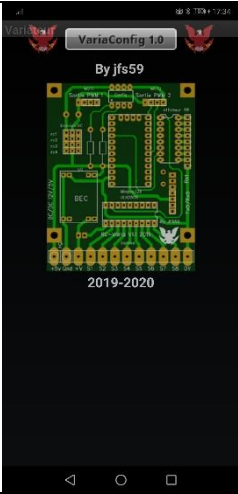



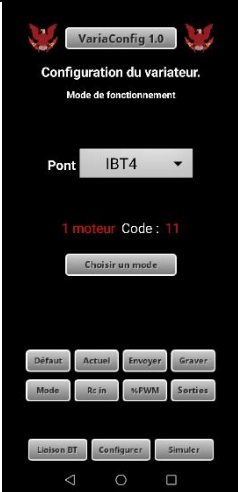
Ce n'est pas la dernière version et elle n'assure donc pas tout le fonctionnement et toutes les simulations.


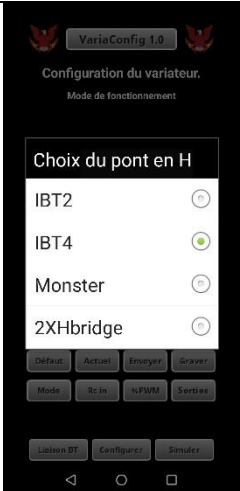
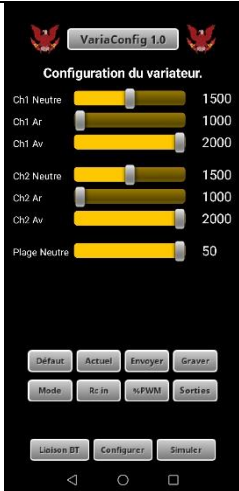
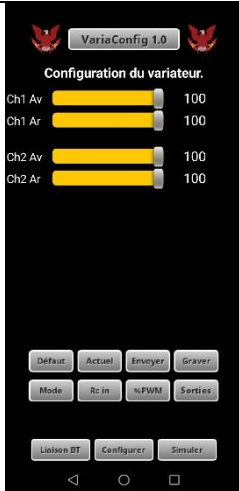




Le programme sera mis à jour en fonction des versions stables qui seront développées ainsi que le sketch Arduino.

Il est préférable qu'ils soient de la même période l'un n'allant pas sans l'autre.


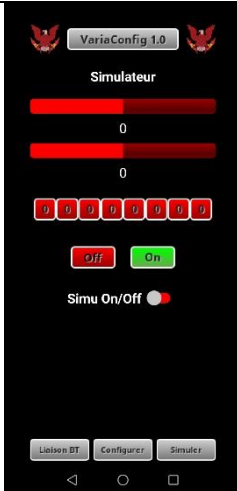

### Sont en développement :

- Gestion des sons mp3.
- Support des différents ponts en H.
- Vitesses différentielles des moteurs en fonction du gouvernail.
- Résolution de bug de communication Bluetooth.

			
<p><b><u>Association Bluetooth : Recherche.</u></b></p> <p>Aller dans paramètres Bluetooth rechercher les appareils disponibles et cliquer sur le JDY-31 une fenêtre d'association va apparaitre.</p>	<p><b><u>Association Bluetooth : Code.</u></b></p> <p>Le périphérique Bluetooth demande une confirmation pour s'associer. Le code est 1234. Ne pas cocher les cases supplémentaires.</p>	<p><b><u>Association Bluetooth : Association</u></b></p> <p>Une fois associé le périphérique reste dans la liste jusqu'à suppression. Il ne faut donc l'associer normalement qu'une seule fois.</p>	<p><b><u>Programme Variateur : Ecran de démarrage</u></b></p> <p>Lancer le programme variateur un splash screen apparait le temps du démarrage.</p>
			
<p><b><u>Page de connexion :</u></b></p> <p>La page connexion apparait. Il faut se connecter avant de continuer sinon vous avez un message disant que le Bluetooth n'est pas connecté.</p>	<p><b><u>Choix du périphérique Bluetooth :</u></b></p> <p>Normalement le JDY 31 est dans la liste le choisir. Le téléphone va essayer de se connecter (Cela peut prendre un certain temps)</p>	<p><b><u>Connexion :</u></b></p> <p>La page de connexion apparait et un message de version est affiché. Les autres pages sont maintenant disponibles.</p>	<p><b><u>Page de mode.</u></b></p> <p>Affiche le mode de fonctionnement du variateur ainsi que le hardware à utiliser. Permet d'activer le mode mp3.</p>

			
<p><b><u>Choix du mode :</u></b></p> <p>Permet de choisir un mode de fonctionnement dans les modes disponibles.</p>	<p><b><u>Choix du matériel :</u></b></p> <p>Permet de choisir le pont en H utilisé par le variateur. (Seul IBT4 fonctionne pour l’instant)</p>	<p><b><u>Paramètres RC :</u></b></p> <p>Permet de choisir les limites du signal RC pour adapter les valeurs.</p>	<p><b><u>Paramètres variateur :</u></b></p> <p>Permet de choisir les limites de la tension de sortie en % de batterie d’alimentation du pont en H.</p>
			
<p><b><u>Paramètres Sorties :</u></b></p> <p>Permet de choisir l’action sur les sorties (monostable, bistable) les seuils sont pour l’instant non modifiables.</p>	<p><b><u>Valeurs par défaut :</u></b></p> <p>Permet de recharger les valeurs par défaut du variateur dans le programme.</p>	<p><b><u>Valeurs actuelles :</u></b></p> <p>Permet de recharger les valeurs présentent dans le variateur.</p>	<p><b><u>Envoyer la configuration :</u></b></p> <p>Permet de charger les valeurs du programme dans le variateur. Elles sont effectives de suite mais ne sont pas sauvegardées. Retour au valeurs sauvegardées au redémarrage.</p>



				
<p><b><u>Enregistrer la configuration :</u></b></p> <p>Permet de graver les valeurs du programme dans le variateur (EEPROM). Elles deviennent les valeurs de démarrage. Le variateur redémarre automatiquement suite à la sauvegarde.</p>		<p><b><u>Simulateur :</u></b></p> <p>Permet de modifier les valeurs en sortie des ponts en H. Et de modifier les valeurs des sorties. Le mode simulateur surpasse les valeurs de la radiocommande.</p>		
				
			<p><b><u>Page About :</u></b></p> <p>Elle contient les infos sur le programme.</p>	

# Structure et extraits du programme Arduino

Le code complet ne sera pas publié par contre je laisse libre accès au fichier compilé et donc au fichier hexadécimal à programmer avec un USBasp. (Les initiés comprendront la terminologie) de même je donne libre accès au BOOLOADER modifié. Je peux sous certaines conditions faire pour vous la mise à jour.

## Variables de configurations :

```
struct __attribute__((packed)) RC_Config {
    unsigned long Flag;
    byte STRUCT_VERSION;
    byte ModeFonctionnement;
    unsigned int Voie_1_Neutre;
    unsigned int Voie_1_Arriere;
    unsigned int Voie_1_Avant;
    unsigned int Voie_2_Neutre;
    unsigned int Voie_2_Arriere;
    unsigned int Voie_2_Avant;
    byte Plage_Neutre;
    byte PourcentAvantV1;
    byte PourcentArriereV1;
    byte PourcentAvantV2;
    byte PourcentArriereV2;
    byte ModeSorties[8];
    int Seuil_Bouton[8];
};
```

## Configuration par défaut :

```
void DefinirConfigDefault() {
    Configuration.Flag = 123456789;
    Configuration.STRUCT_VERSION = 1;
    Configuration.ModeFonctionnement = 11 ;
    Configuration.Voie_1_Neutre = 1500;
    Configuration.Voie_1_Avant = 2000;
    Configuration.Voie_1_Arriere = 1000;
    Configuration.Voie_2_Neutre = 1500;
    Configuration.Voie_2_Avant = 2000;
    Configuration.Voie_2_Arriere = 1000;
    Configuration.Plage_Neutre = 50;
    Configuration.PourcentAvantV1 = 100;
    Configuration.PourcentArriereV1 = 100;
    Configuration.PourcentAvantV2 = 100;
    Configuration.PourcentArriereV2 = 100;
    for ( int n = 0; n < 8; n ++ ) {
        Configuration.ModeSorties[n] = 0 ;
    }
    for ( int n = 0; n < 8; n ++ ) {
        Configuration.Seuil_Bouton[n] = 1500 ;
    }
}
```

```
}
```

### Gestion des entrées RC par interruption :

```
static byte rcOld;    // Prev. states of inputs

volatile unsigned long rcRises[4]; // times of prev. rising edges
volatile unsigned long rcTimes[4]; // recent pulse lengths
volatile unsigned int rcChange=0; // Change-counter
// Be sure to call setup_rcTiming() from setup()

void setup_rcTiming() {
    rcOld = 0;

    pinMode(A0, INPUT_PULLUP); // pin 14, A0, PC0, for pin-change interrupt
    pinMode(A1, INPUT_PULLUP); // pin 15, A1, PC1, for pin-change interrupt
    pinMode(A2, INPUT_PULLUP);
    pinMode(A3, INPUT_PULLUP);

    PCMSK1 |= 0x0F;    // Four-bit mask for four channels
    PCIFR  |= 0x02;    // clear pin-change interrupts if any
    PCICR  |= 0x02;    // enable pin-change interrupts
}

// Define the service routine for PCI vector 1
ISR(PCINT1_vect) {
    byte rcNew = PINC & 15; // masquage 4 bits, A0-A3
    byte changes = rcNew^rcOld; // verif changement bit
    byte channel = 0;
    unsigned long now = micros(); // micros() ok
    while (changes) {
        if ((changes & 1)) { // Did current channel change?
            if ((rcNew & (1<<channel))) { // Check rising edge
                rcRises[channel] = now; // Is rising edge
            } else { // Is falling edge
                rcTimes[channel] = now-rcRises[channel];
            }
        }
        changes >>= 1; // shift out the done bit
        ++channel;
        ++rcChange;
    }
    rcOld = rcNew; // Save new state
}
```

### Message de Boot et trame de configuration Bluetooth : (liaison série 9600 bauds)

<Variateur IBT4 + RC Addon

Version 3.2.57 BT

Mar 19 2020

>

<CmdCfg|1|11|1500|1000|2000|1500|1000|2000|50|100|100|100|100|0|0|0|0|0|0|1500|1500|1500|1500|1500|1500|1500|1500>