## Pilotage des servomoteurs.

Actuellement tous les servomoteurs utilisés en petite robotique fonctionnent sur le même principe. Bien que les fournisseurs ne respectent pas forcément ce qui semble devenir une norme de fait, globalement tous s'approchent de ce qui devient un standard. Ces moteurs intégrant une électronique d'interfaçage et un capteur de position se raccordent avec seulement trois fils :

- Le fil noir ou ***marron*** va à la masse **GND**.
- Le fil central ***rouge*** va au **+5Vcc**. *(1)*
- Le fil blanc ou ***orange*** reçoit le signal de pilotage.

Comme montré sur la Fig.1 la fréquence de répétition standard choisie pour leur pilotage est de 50 Hz, soit une période de 20 mS. C'est la durée **T** de l'impulsion à l'état "1" qui positionne le moteur en absolu par rapport à son amplitude possible de rotation. Mis à part les moteurs sans limitation de rotation angulaire, les plages de débattement angulaire les plus courantes avoisinent les 180°.

***En standard :***

- T = 1 mS : Le moteur se place en position minimale.
- T = 1,5 mS : Le moteur se place en position neutre. *(2)*
- T = 2 mS : Le moteur se place en position maximale.

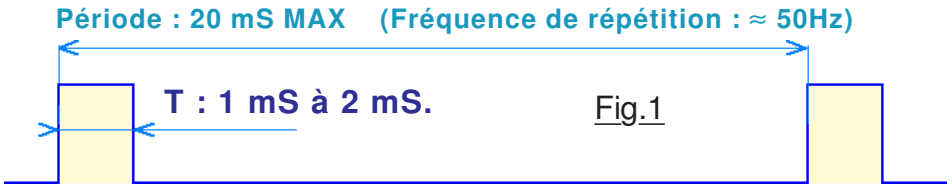



Fig.1

*Attention : La proportion temporelle n'est pas respectée.*

*(1)* Les moteurs à courant continu génèrent des appels de courant importants quand ils démarrent. Chaque surintensité transitoire corrompt la tension d'alimentation. Pour qu'Arduino ne soit pas perturbé, il faut impérativement qu'il reçoive son **+5Vcc** d'une ***alimentation indépendante de celle des moteurs***. En ravanche **GND** ***est commun aux deux sources d'alimentation***.

*(2)* La position neutre correspond à la position moyenne qui pour la course de 180° se trouve au centre à 90°.



Fiche n° 1

## Petit module LASER.

### Caractéristiques techniques du petit module LASER :

- C'est une simple diode LASER associée à une résistance de 103 Ω montée en série pour limiter le courant.
- Longueur d'onde 650 nm. *(Couleur rubis)*
- Puissance lumineuse 2 à 5 mW.
- Le +5Vcc peut être appliqué directement sur la broche d'alimentation, mais pour augmenter la durée de vie de la diode LASER il est possible d'insérer une résistance. Le tableau proposé ci-dessous précise le courant d'alimentation en fonction de **R**, la tension d'alimentation étant de +5Vcc.
- Le courant maximal consommé est compatible avec la sortance d'une broche binaire de l'ATmega328. Il est parfaitement possible de moduler la luminosité par PWM, la fréquence de 490Hz de découpage étant largement assez élevé pour donner l'impression d'une clarté constante.

| R | 0 | 22 Ω | 47 Ω | 82 Ω | 100 Ω |
|---|---|---|---|---|---|
| I | 22 mA | 19 mA | 17 mA | 14 mA | 14 mA |

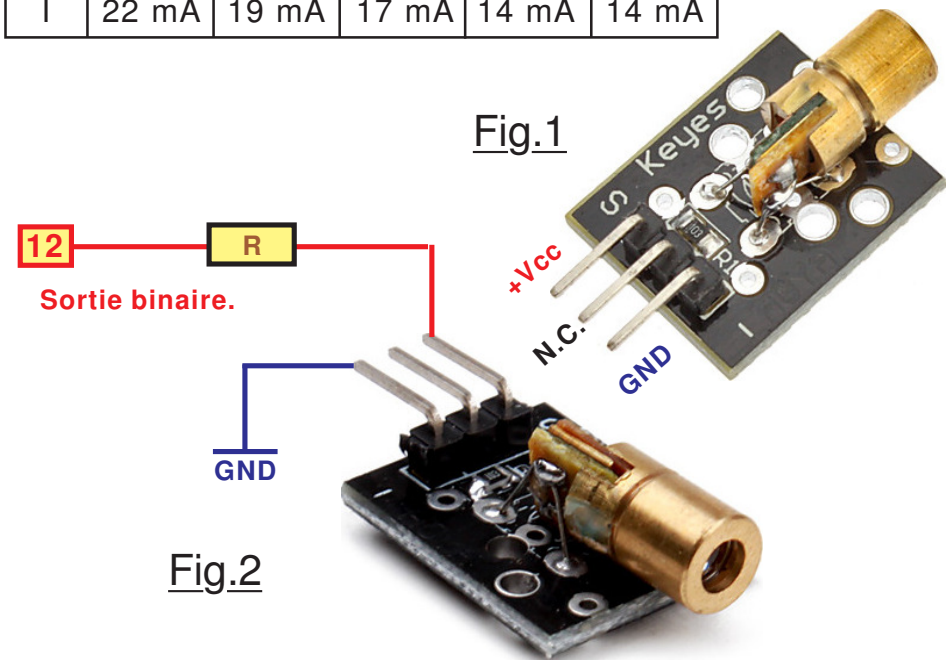



Fig.2

# *Répartition des moteurs sur l'interface.*







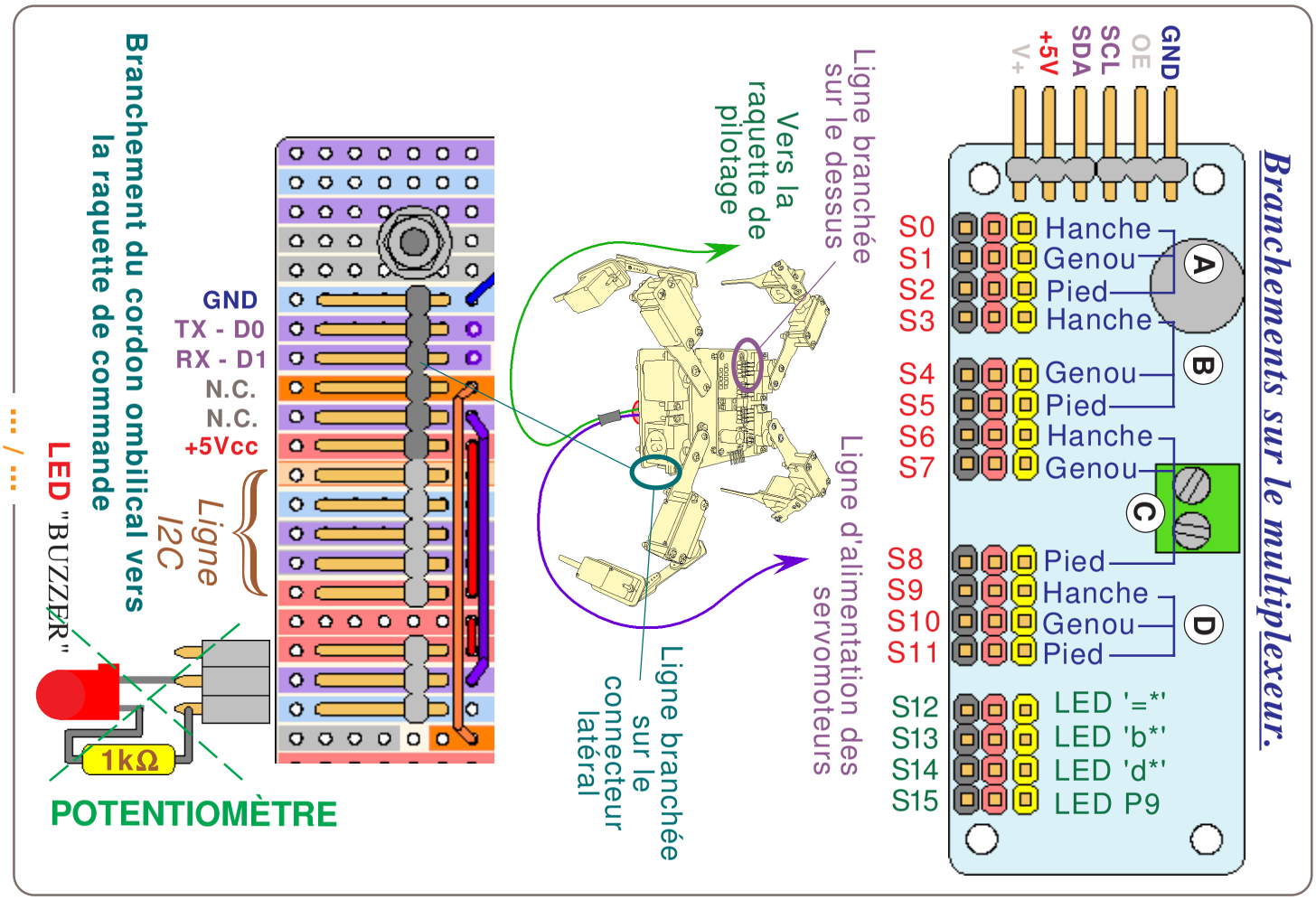

## *Repérage et caractéristiques des articulations.*



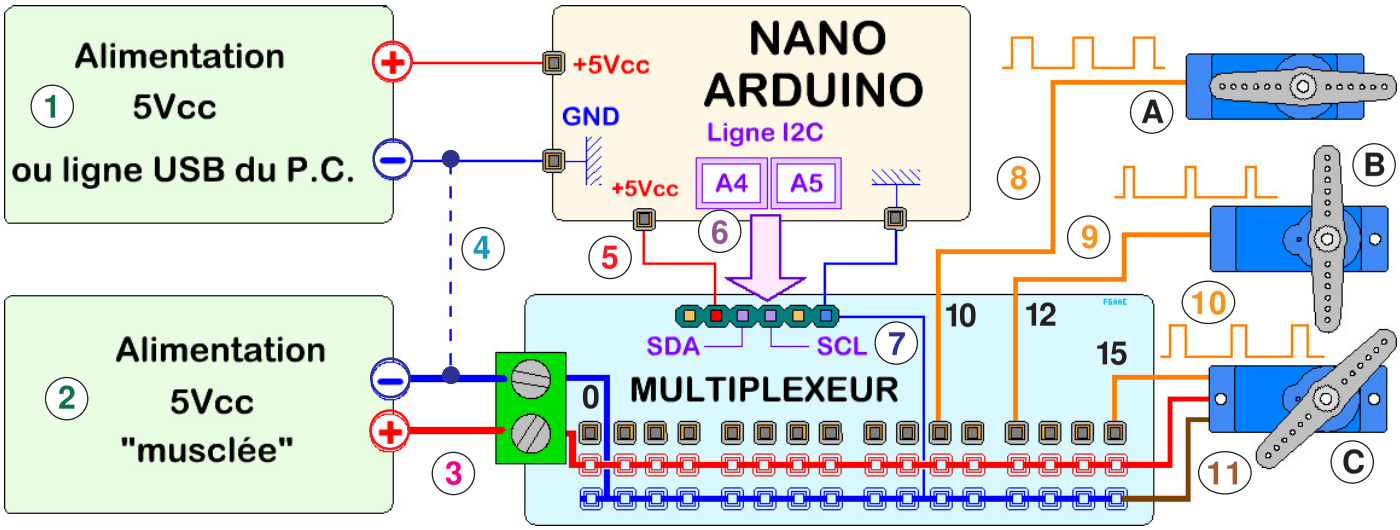

**Jambe robotisée.**

Repéré en vert les membres principaux.
Repéré en bleu les articulations.

| Jambe | Articulation | Traçabilité | Sortie | Butée - | Centrage | Butée + | Moteur |
|---|---|---|---|---|---|---|---|
| A | Hanche | | | | | | 1 |
| | Genou | | | | | | 2 |
| | Pied | | | | | | 3 |
| B | Hanche | | | | | | 4 |
| | Genou | | | | | | 5 |
| | Pied | | | | | | 6 |
| C | Hanche | | | | | | 7 |
| | Genou | | | | | | 8 |
| | Pied | | | | | | 9 |
| D | Hanche | | | | | | 10 |
| | Genou | | | | | | 11 |
| | Pied | | | | | | 12 |




Fiche n° 3




En **1** la ligne USB du P.C. alimente la carte Arduino NANO qui ne consomme qu'une énergie dérisoire. On utilise des fils de faible section. En **2** l'alimentation musclée en "gros fils" **3** va directement au bornier de puissance du "multiplexeur". Logiquement en **4** il faudrait établir un lien entre les deux masses de référence. Ce n'est pas indispensable, car en **7** la liaison est établie en interne sur la carte électronique. Tous les moteurs branchés seront en parallèle comme en **11** sur les deux lignes d'alimentation de puissance. Chaque ligne orange d'un moteur va sur la sortie **0** à **15** de pilotage qui lui est réservée. L'électronique locale est alimentée par une ligne **5** à part issue de la carte Arduino. Le pilotage de la carte "multiplexeur" se fait par des consignes envoyée sur la **ligne I2C** en **6** et respecte les protocoles du circuit intégré soudé sur le circuit imprimé. Recevant une consigne cohérente, le circuit interne va alors fournir au servomoteur un signal PWM idoine sur une ligne telle que **8**, **9** ou **10** pour le positionnement angulaire de l'articulation concernée.

## Méthodes de la bibliothèque Adafruit_PWMServoDriver.h.

Spécifique au multiplexeur PCA9685 à 16 canaux cette bibliothèque permet facilement de gérer chaque moteur indépendamment l'un de l'autre. L'effet d'une commande reste effectif jusqu'à une nouvelle consigne pour le canal concerné. Toute utilisation de cette bibliothèque doit être précédée de la déclaration :

```
#include <Adafruit_PWMServoDriver.h>
Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();
```

### *Fréquence du signal généré sur les 16 canaux.*

```
pwm.begin();
```

Cette instruction doit être placée en premier dans void **setup**() et permet de générer une instance du module PCA9685.

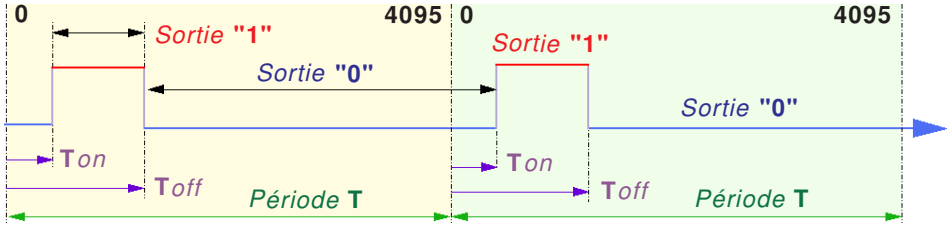
### *Fréquence du signal généré sur les 16 canaux.*

```
pwm.setPWMFreq(Fréquence);
```

Cette instruction doit être placée dans void **setup**() et définit la fréquence des signaux PWM générés sur les sorties. La valeur sera comprise entre 40 et 1000, mais en standard adopter 50Hz. *(La période **T** fait alors 20mS pour des servomoteurs.)*

### *Principe de la génération PWM.*

La génération PWM est basée sur un compteur à 12 BITs qui fonctionne à une cadence définie par pwm.setPWMFreq() et qui




en détermine la période **T**. Le signal présentera une période identique sur toutes les sorties d'un même module. L'instruction qui conditionne une sortie contient deux paramètre *Ton* et *Toff* qui permettent à notre guise de définir la durée à l'état "1". Ce sont les valeurs du compteur qui déclenchent les changements d'état :

*Sortie* "**1**" = *Toff* - *Ton*.
*Sortie* "**0**" = *Période* **T** - *Sortie* "**1**".

Le rapport cyclique et la durée des deux états est donc directement fonction de la fréquence de répétition programmée.

... / ...



## Méthodes de la bibliothèque Adafruit_PWMServoDriver.h.

Fondamentalement le module PCA9685 est créé pour générer de la PWM dans des applications quelconques. C'est la raison pour laquelle on peut librement définir la fréquence de répétition et le rapport cyclique. On peut par exemple piloter "analogiquement" des éclairages, des résistances chauffantes etc. *Les exemples et la formule donnée ci-dessous sont donc restrictives à une période imposée "arbitrairement" à 20mS pour piloter des servomoteurs.*

### *Largeur de l'impulsion "positive" générée.*

```
pwm.setPWM(Num_sortie, Ton, Toff)
```

Cette instruction engendre sur la sortie Num_moteur un signal PWM à la Fréquence prédéfinie dont la durée de l'état "1" est définie par les valeurs Toff - Ton sur une "portée" 4095.

Pour s'affranchir de la différence, Ton ***sera généralement égal à 0***.
Compteur = 4095 pour 20mS soit 20000µS.
On a un comptage pour 20000 / 4095 soit toutes les 4,884µS.

Toff à indiquer = *Sortie* "**1**" / 4,884 *(Sortie "**1**" désirée en µS.)*

*Exemple : Sortie "1" désirée = 2272µS.*
*Toff = 2272 / 4,884 = 465 ⇒ pwm.setPWM(Num_Sortie,0,465);*

### *Pilotage en degrés des servomoteurs.*

Chaque moteur peut en principe balayer un angle de 180°. Toutefois la dispersion de caractéristique fait que d'un moteur à l'autre la position adoptée n'est pas rigoureusement identique pour un même signal. Par exemple le moteur testé présente les limites suivantes :
-90° pour 2272µS soit Sup = 465 valeur nommée Tmini.
+90° pour 667µS soit Sup = 136 valeur nommée Tmaxi.
L'instruction qui transpose l'angle désiré en durée PWM est :

```
Pulse = map(Angle, -90, +90, Tmini, Tmaxi);
```

**Ligne I2C et adressage du module :** La bibliothèque impose pour la ligne I2C le signal **SDA** sur **A4** et le signal **SLC** sur **A5**. L'adressage par défaut sur le matériel est en 0x40. On peut à la demande, ce qui sera obligatoire si on chaîne plusieurs modules, modifier l'adresse matériellement par établissement de contacts sur des ponts disponibles sur le circuit imprimé. *(Voir documentation.)*

... / ...

# BUS série au standard I2C. (1/2)

Développé initialement par Philips en 1982, le bus I2C s'est largement imposé dans le domaine des microprocesseur, microcontrôleurs et applications industrielles diverses. Sa désignation dérive de Inter-Integrated Circuit. Il fut à l'origine conçu pour des applications de domotique et d'électronique domestique.
La norme I2C est basée sur un ***bus série synchrone bidirectionnel*** fonctionnant en "half-duplex", où plusieurs périphériques maîtres ou esclaves peuvent communiquer entre eux. Les dialogues ont toujours lieu entre un seul maître et un ou tous les esclavesprésents et l'échange de données et toujours déclenché à l'initiative du maître. *(Jamais de maître à maître ou d'esclave à esclave.)*

## ➢ Constitution matérielle du bus I2C.

Outre une masse commune GND la ligne n'utilise que deux fils :
- **SDA** *(**S**erial **Da**ta Line)* : Ligne de données bidirectionnelle,
- **SCL** *(**S**erial **Cl**ock Line)* : Ligne d'horloge pour la synchronisation également de type bidirectionnel.

Les deux lignes sont maintenues à l'état logique "1" par un niveau de tension +VDD à travers des résistances de forçage. *(Pull-Up.)*
Dans le standard I2C le nombre maximal de périphériques est limité à 128 par le nombre d'adresses disponibles, 7 Bits d'adressage et un Bit R/W. *(Lecture ou Écriture.)* Bien qu'une foule de combinaisons complexes soit possible, dans le cas d'Arduino c'est la carte ATmega328 qui sera toujours le Maître et les modules périphériques les esclaves. Traditionnellement si le programme d'application doit intégrer une ligne I2C, comme représenté sur la Fig.1 ci-dessous ce ***sont* A4 *et* A5** qui ***sont respectivement affectées à* SDA *et* SCL**. Les bibliothèques qui accompagnent les modules du commerce sont basées sur ce schéma.

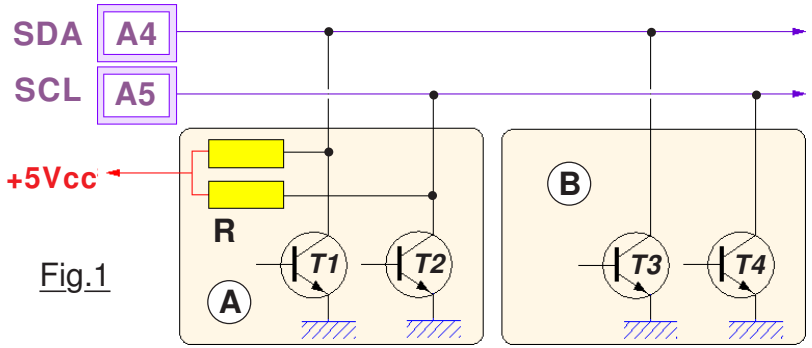



Fig.1

... / ...



# BUS série au standard I2C. (2/2)

## ➢ Signaux échangés sur un bus I2C.

Le niveau HIGH ou LOW de la ligne **SDA** doit être maintenu stable pendant le niveau HIGH sur la ligne **SCL** servant à déclencher la lecture successive des bits du protocole de dialogue. *(Voir la Fig.2)*

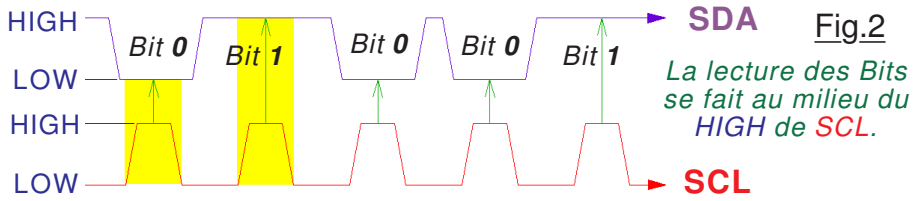



Fig.2

*La lecture des Bits se fait au milieu du HIGH de SCL.*

Comme montré sur la Fig.1 les équipements sont connectés au bus par des électroniques de type drain ouvert. *(Ou collecteur ouvert.)* Fonctionnant en ET câblés deux périphériques tels que **A** et **B** peuvent "parler" simultanément. Dans ce cas un état logique "0" "écrase" un état logique "1". Pour caractériser ce genre d'incident potentiel sur un bus I2C on utilise le vocable :
- L'état logique "0" LOW est un état dominant,
- L'état logique "1" HIGH est un état récessif.

Lorsque le bus n'est pas utilisé, il est forcé niveau haut par les résistances telles que **R** de l'un des modules connectés. Il suffit d'un seul rappel à +5Vcc pour que la ligne fonctionne. Si plusieurs modules sont pourvus de résistances de forçage, le courant qui devra être drainé par l'ATmega328 et les électroniques branchées sera plus important mais reste généralement faible car le nombre de périphériques est classiquement faible pour des applications ordinaires.
Pour les modules dédiés à Arduino les vitesses de transmission sont généralement comprises entre 100kb/s et 400kb/s.

## ➢ Les bibliothèques de programme.

Pour simplifier le travail des programmeurs et se positionner sur le marché, les concepteurs de modules électroniques dialoguant en I2C fournissent des bibliothèques dédiées à leurs produits. Outre les "library" propres à chaque référence commerciale, la bibliothèque Wire.h est spécialisée pour gérer sur carte Arduino les protocoles I2C/TWI et devra parfois accompagner celle qui accompagne un module électronique spécialisé. C'est elle qui impose l'usage de **A4** et **A5**.

... / ...

# SCHÉMAS ÉLECTRONIQUES.

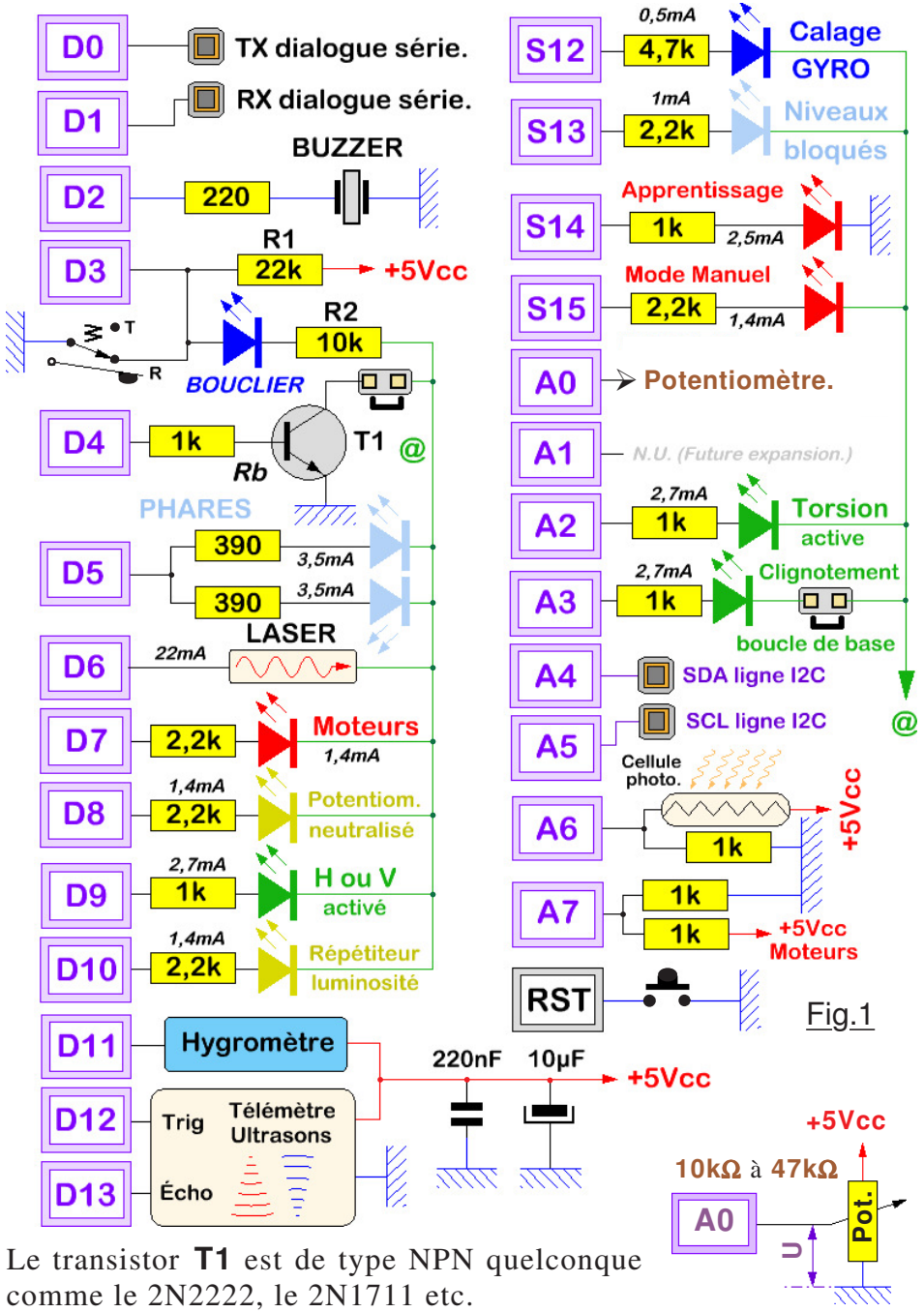



Fig.1

Le transistor **T1** est de type NPN quelconque comme le 2N2222, le 2N1711 etc.

... / ...



## Affectation des Entrées / Sorties.

D0 : *Dialogue série avec le moniteur de l'IDE.*
D1 : *Dialogue série avec le moniteur de l'IDE.*
D2 : Pilotage du BUZZER.
≈ D3 : Détection du contact du bouclier avec le sol.
D4 : Coupure de masse de toutes les LED. *(Pilotage de la base du transistor de commutation.)*
≈ D5 : Éclairage analogique *(PWM)* des phares.
≈ D6 : Pilotage du LASER.
D7 : Pilotage LED témoin Moteurs figés.
D8 : Pilotage LED témoin Potentiomètre neutralisé.
≈ D9 : Pilotage LED témoin mode H ou V.
≈ D10 : Pilote la LED répétiteur du CAN potentiomètre.
≈ D11 : Capteur Humidistance et de température.
D12 : Transmetteur ultrasons.
D13 : Récepteur ultrasons.

---

A0 : Sur la version P.C : Commande analogique.
A1 : Libre pour F.E. car non utilisée.
A2 : Pilotage LED de torsion active.
A3 : Clignotement de la LED de "boucle de PGM active".
A4 : SDA pour la ligne I2C.
A5 : SCL pour la ligne I2C.
A6 : Mesure de la lumière pour le luxmètre.
A7 : Mesure de la tension d'alimentation des moteurs.

---

Sortie S12 Multiplexeur : LED de la commande '=*' armée.
*(Mémorisation du Gyroscope sur le Lacet actuel.)*
Sortie S13 Multiplexeur : LED de l'option 'B*' active.
*(Niveau éclairages des phares et du LASER Bloqués.)*
Sortie S14 Multiplexeur : LED du mode 'D*' actif. .....
*(Enregistrement d'une suite de déplacements en cours.)*
Sortie S15 Multiplexeur : LED du mode "P9n*" en cours.
*(Pilotage des moteurs manuellement et individuellement.)*

... / ...

# Circuit imprimé principal. (1/3)

ATTENTION : Sur ce dessin le circuit imprimé est vu coté composants avec la représentation des pistes cuivrées vues par transparence comme si la plaque isolante était translucide.

**Ensemble complet entièrement achevé.**

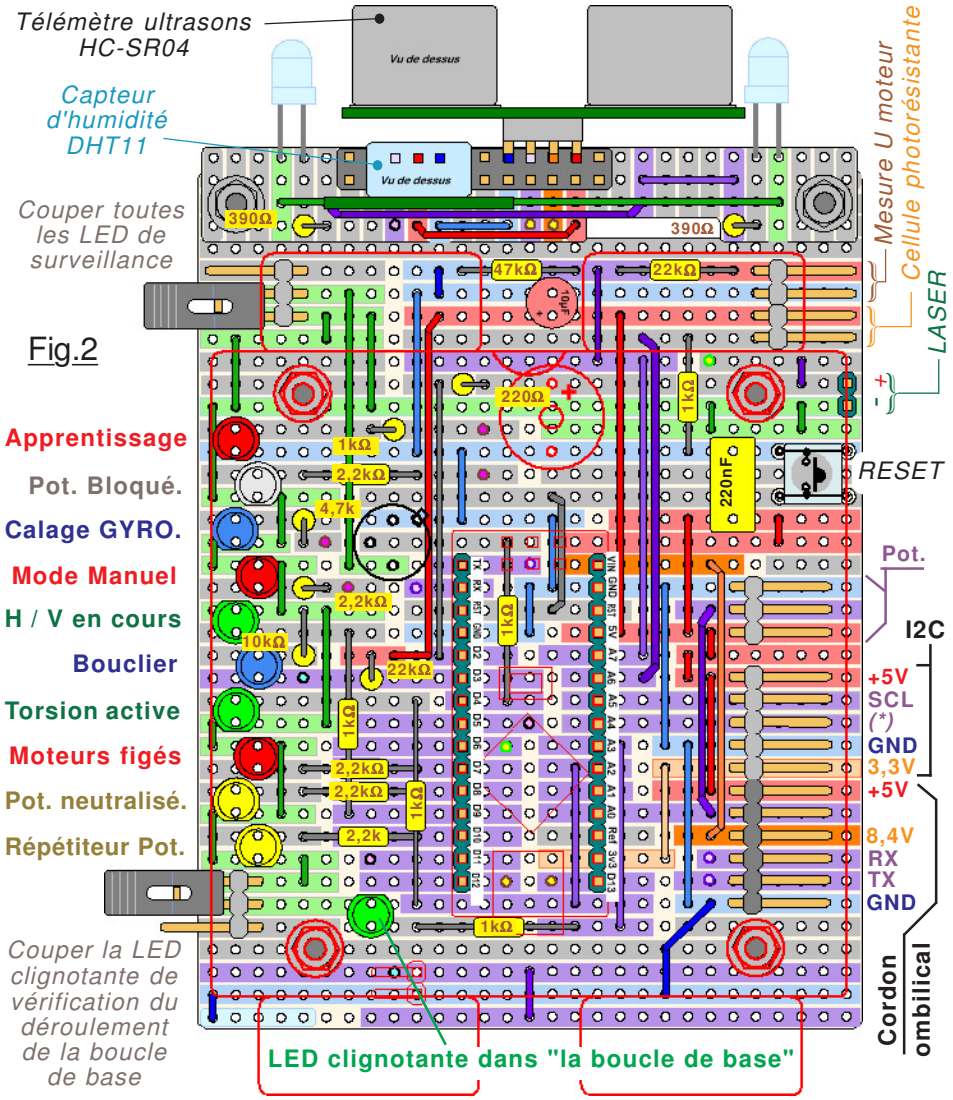



Fig.2

- En vert sont représentées les pistes de la ligne @.
- La piste (*) sur la ligne I2C n'est pas utilisée.

*POT : Potentiomètre linéaire*

... / ...



# Circuit imprimé principal. (3/3)

**Ensemble complet entièrement achevé.**

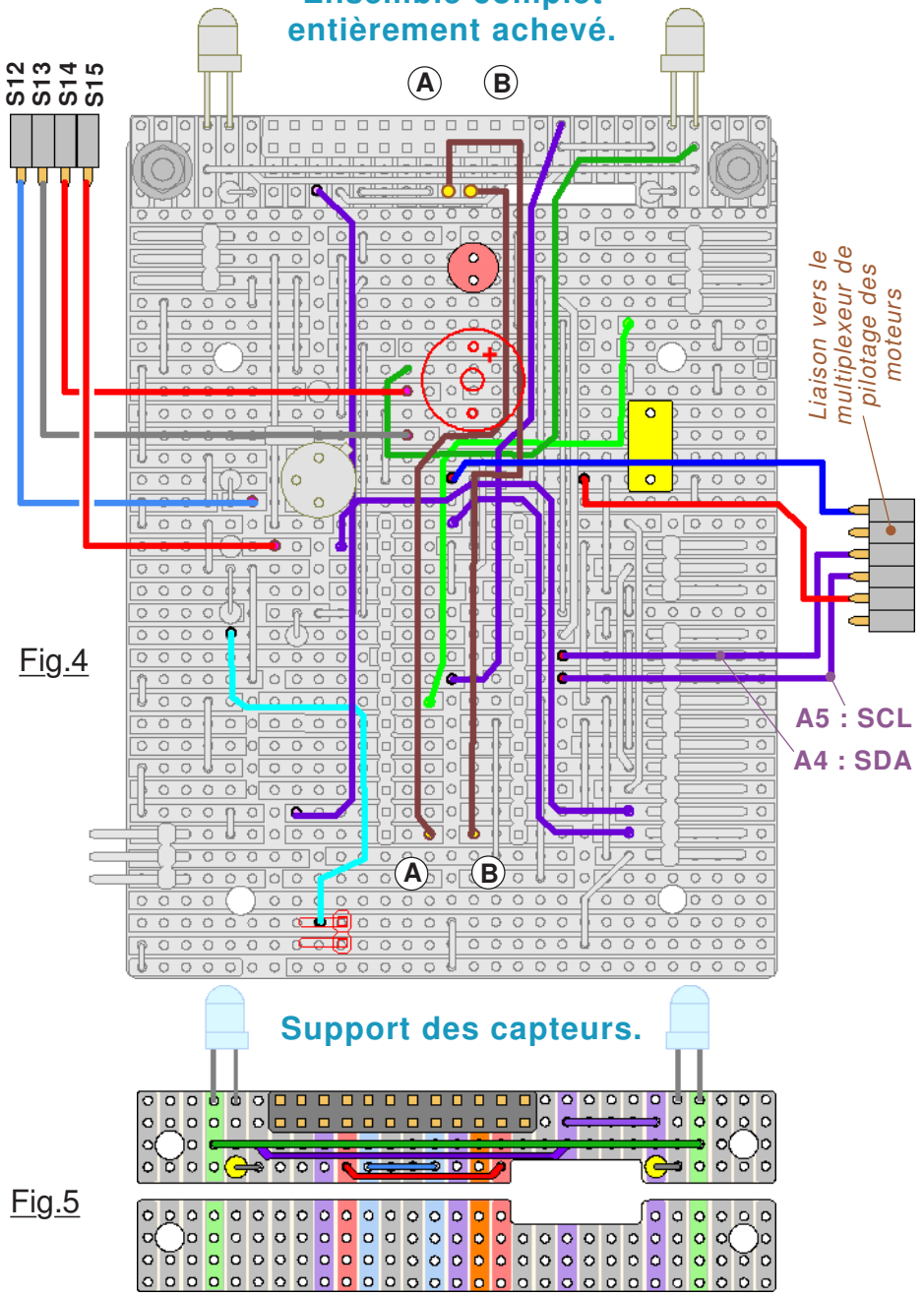



Fig.4

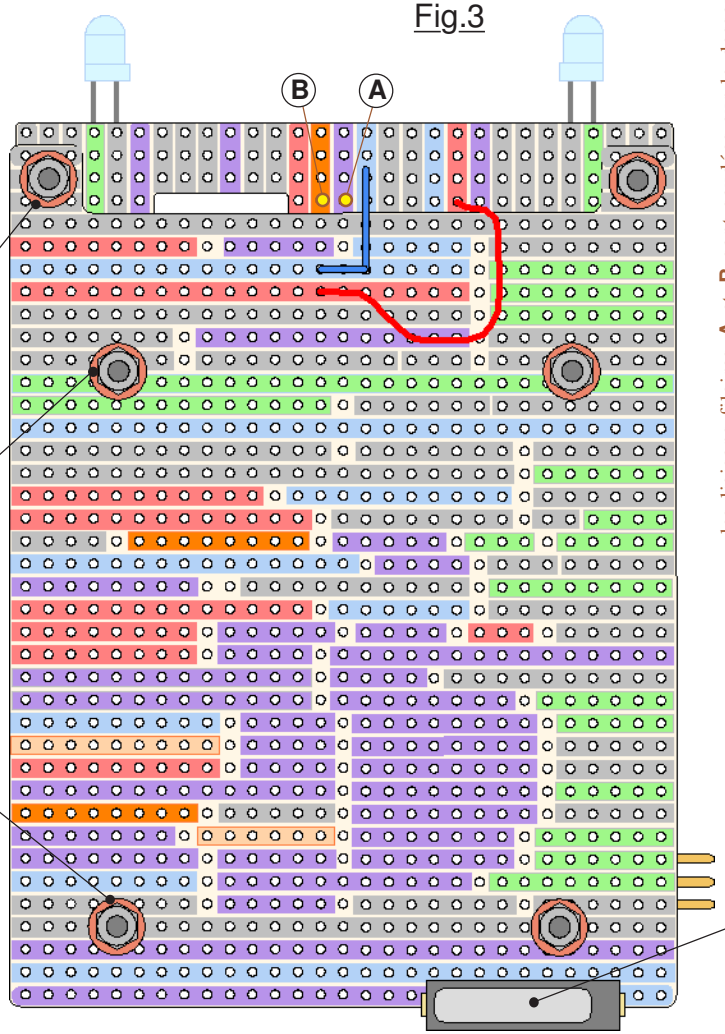
Fig.5

... / ...

# *Circuit imprimé principal.* (2/3)

Sans que ce ne soit systématique, d'une façon générale :
Le bleu représente des pistes à la masse GND,
Le rouge le +5Vcc, le orange clair le +3.3Vcc,
Le orange foncé le +9V issu d'un éventuel accumulateur déporté,
Le vert la masse coupée par le transistor NPN notée @.
Le violet des Entrées / Sorties sur la carte Arduino Nano,
Le gris des liaisons "banales".

Fig.3

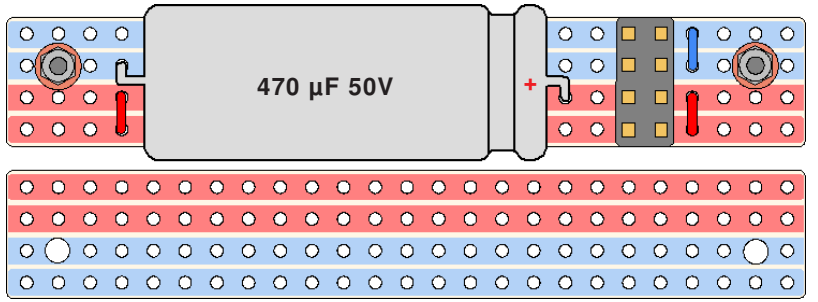



**Circuit entièrement câblé vu coté pistes cuivrées.**

Fiche n° 8

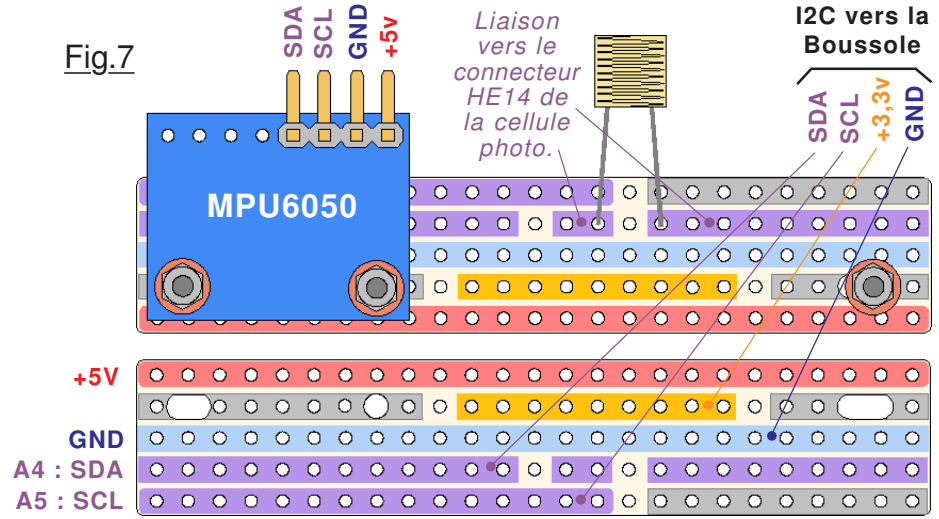
# *Circuits imprimés auxiliaires.*

Fig.6 **Support du condensateur "réservoir".**




**Support du capteur photorésistant et de la centrale gyroscopique.** *(Liaisons I2C.)*

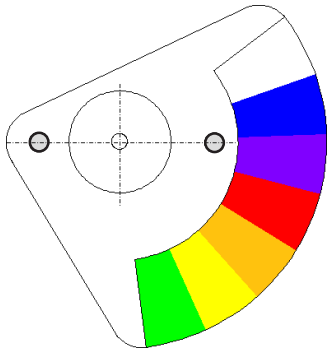
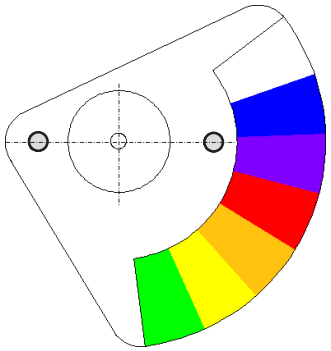
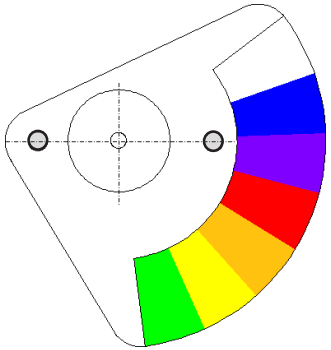
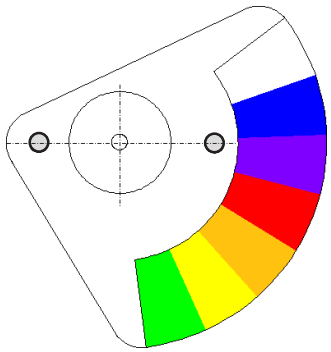
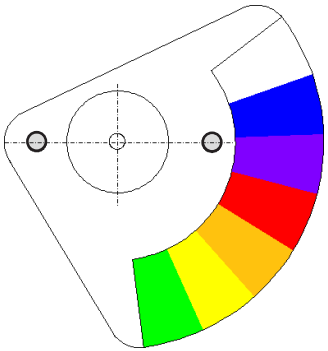
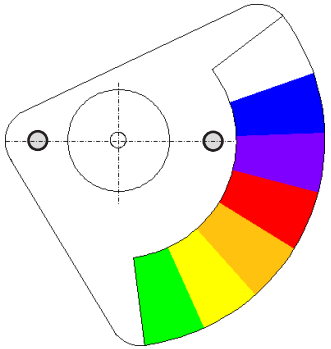
Fig.7




ATTENTION : Il faut impérativement intercaler des rondelles isolantes coté cuivre pour ne pas que les écrous métalliques ne provoquent des courts-circuits entre les pistes conductrices.

ATTENTION : En fonction du module MPU6050 approvisionné, il n'est pas obligatoire que le circuit intégré ne soit implanté de façon identique. Les axes de roulis et de tangage peuvent donc ne pas présenter la même orientation. Il faudra étudier la position du module sur le circuit imprimé en fonction des axes de référence.

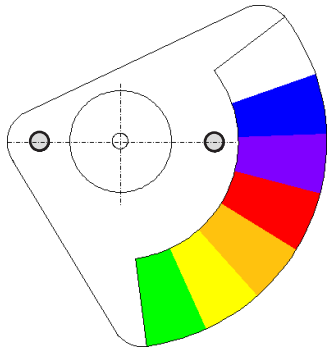
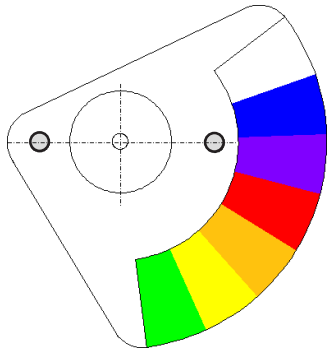
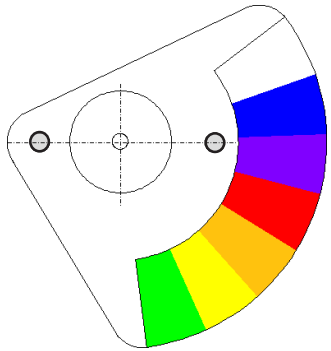
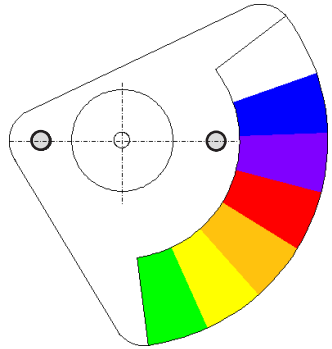
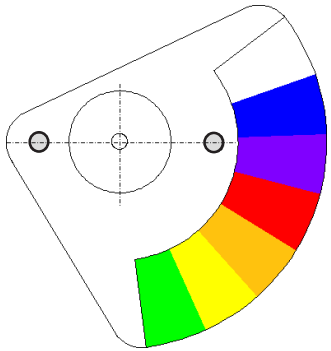
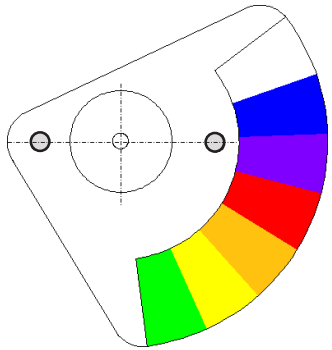
## ***Filtres colorimétriques.*** *(1/2)*

… / …

Fiche n° 9

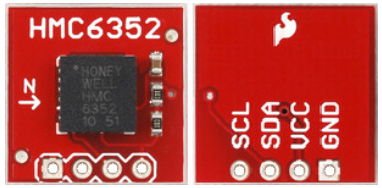
## ***Filtres colorimétriques.*** *(2/2)*

… / …

# ***BOUSOLE statique HMC6352.*** *(1/2)*

Comme tous les modules qui utilisent une SPI pour réaliser le dialogue avec Arduino, ce petit circuit est extrêmement facile à mettre en œuvre. Le schéma du petit module est donné en Fig.1 avec le circuit imprimé représenté par dessus. La flèche verte sur le dessin représente la direction du "Nord magnétique" de cette boussole statique. La petite flèche notée **N** sur le circuit imprimé définit ce pôle du circuit intégré. Quand cette flèche est orientée vers le Nord magnétique local, le HMC6352 retourne la grandeur 0° pour la valeur du Cap.




## Caractéristiques techniques du module HMC6352 :

- Tension de fonctionnement : 2,7Vcc à 5,2Vcc.
- Consommation 2mA. Maximum 10mA sous 5Vcc.
- Compas avec donnée de sortie en valeur décimale pour le CAP.
- Intègre deux axes magnétiques complets.
- Logiciel local de traitement d'acquisition, de calibrage et de calcul de CAP inclus au circuit intégré.
- Interface série de type I²C. ***(Adresses : 42HEX et 43 HEX)***
- Fonctionne en esclave. Fréquence d'horloge possible 100kHz.
- Compensé en température.
- Résolution pour le CAP calculé 0,5° avec une répétabilité de 1°.
- Peut être utilisé dans un environnement de champs magnétiques importants.

*ATTENTION : Bien que réputé protégé contre les champs magnétiques importants, sa mise en présence d'un aimant de gestion de la tête d'un disque dur, (Aimant très puissants) a engendré un mauvais fonctionnement du circuit. Pour retrouver sa sensibilité il a fallu soumettre la "puce" à un champ inverse, puis attendre plusieurs heures que la "rémanence" magnétique se dissipe.*

- Plage de champ magnétique ±2Oe.

*L'oersted (Symbole Oe) est l'unité ÉLECTROMAGNÉTIQUE CGS à trois dimensions d'excitation magnétique ou de champ magnétique. L'oersted, nommé ainsi en l'honneur de Hans Christian Ørsted.*

... / ...

Fiche n°10

*(2/2)*

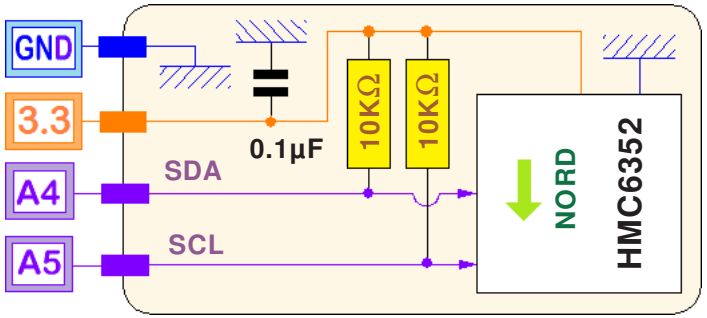



Fig.1

Outre le condensateur de découplage sur l'alimentation en 3,3Vcc on peut noter sur la Fig.1 que les deux lignes de la SPI sont forcées à l'état logique "1" par des résistances de 10 kΩ incluses. C'est un impératif s'il n'y a pas d'autre module I²C en ligne, car ce type de périphérique est à "collecteur ouvert" pour permettre le multiplexage de plusieurs esclaves. En standard pour du dialogue I2C sur Arduino on utilise les deux broches analogiques **A4** pour des données **SDA** et **A5** pour le signal de synchronisation **SCL**.

**ATTENTION :** Le module boussole HMC6352 ne fournit une information de ***Cap magnétique correcte que s'il se trouve dans une attitude parfaitement horizontale***. Il importe donc de veiller à cette contrainte lors des expérimentations et pour son assemblage sur la structure de l'application. Conformément à tout dispositif influencé par un champ magnétique, il donnera une information relative aux "lignes de champ" locales. ***Toute masse magnétique ou aimant dans son environnement faussera l'information relative au champs magnétique terrestre***.

Comme pour tout compas magnétique, l'utilisation d'un tel dispositif doit s'accompagner d'un graphe donnant les ***courbes de correction*** angulaire pour tenir compte des perturbations engendrées par la structure du mobile sur lequel est employé ce module. Le graphe de correction pour le prototype est représenté sur la Fiche n°12.

Initialement, pour pouvoir tester ce petit module la bibliothèque spécialisée hmc6352.h avait été intégrée dans l'environnement de développement. Mais le programme actuel d'utilisation sur la sonde n'en a pas besoin, il faut juste inclure les routines de gestion de la ligne I²C avec la déclaration :

```
#include <Wire.h>
```

... / ...

# *Module ultrasons HC-SRO4.*

## Caractéristiques techniques du module HC-SR04 :

- Alimentation : 5Vcc.
- Consommation en utilisation : 15 mA.
- Gamme de distance : 2 cm à 4 m. *(Limité ici 2,5m par le byte.)*
- Résolution : 3 mm.
- Angle de mesure : < 15°.

## Mise en œuvre :

La Fig.3 montre les branchements à effectuer. Il faut envoyer une impulsion état logique "1" d'au moins 10 µs sur la broche Trig Input pour déclencher une mesure. *(Voir Fig.1)* En retour la sortie Output ou *(Echo)*, va restituer une impulsion d'environ + 5v dont la durée est proportionnelle à la distance si le module détecte un objet. Afin de pouvoir calculer la distance exprimée en cm, on utilisera la formule suivante :

Distance en cm = (Durée de l'impulsion Echo en µs) / 58

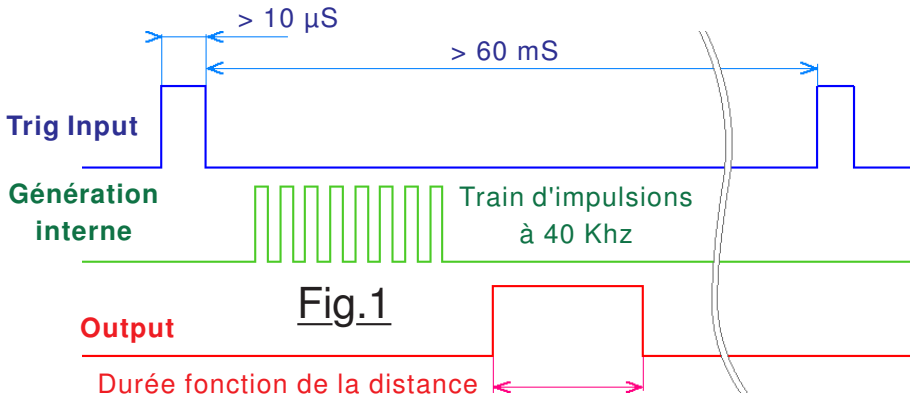
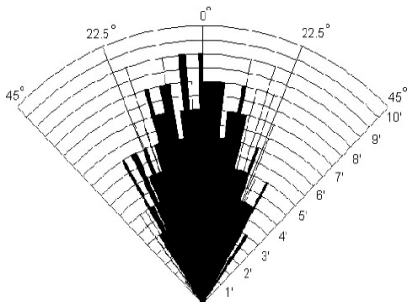



Fig.1




Les distances sont en pouces

**Diagramme polaire**

Fig.2

... / ...



## Exemple de programme :

```
#define TX_Pulse_sonard 12
#define Echo 13
byte Distance;
int  Lecture_Echo;

void setup() {
pinMode(Trig, OUTPUT); digitalWrite(Trig, LOW);
pinMode(Echo, INPUT); }

void Telemetre_ultrasons() {
  // Déclencher une mesure :
  digitalWrite(TX_Pulse_sonard, HIGH);
  delayMicroseconds(10);
  digitalWrite(TX_Pulse_sonard, LOW);
  // Analyser le retour sur la ligne de réception :
 Lecture_Echo = pulseIn(Echo, HIGH);
 Distance = Lecture_Echo / 58;}
```

La procédure Telemetre_ultrasons effectue une mesure et préserve le résultat dans la variable Distance.

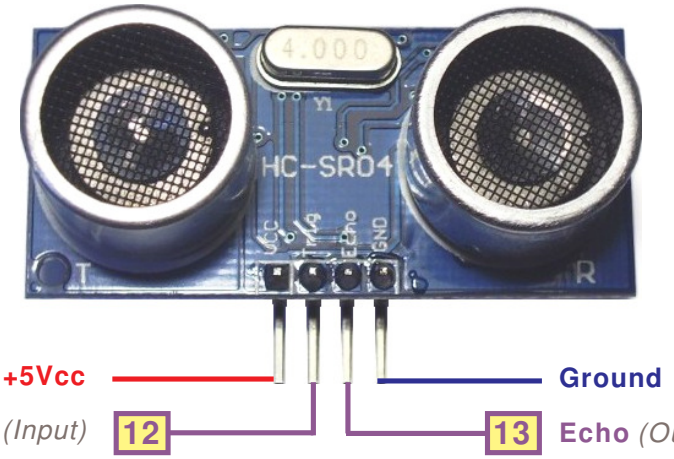



Fig.3

Comme le montre la Fig.2 le capteur à ultrasons HC-SR04 présente un angle d'ouverture trop important *(Cône d'environ 45 degrés)* qui interdit une analyse fine et ne permet pas réellement de différencier la forme des objets situés dans le champ du balayage.

... / ...

# Capteur Température / Humidité DHT11.

Montré sur la Fig.1, ce module de mesure est constitué d'un senseur de température à base d'une thermistance à coefficient négatif de température et d'un capteur d'humidité résistif. Un microcontrôleur intégré s'occupe de faire les mesures, de les convertir et de les transmettre. Le DHT11 fait partie de ces circuits numériques qui dialoguent au moyen d'une seule ligne série bidirectionnelle "à collecteur ouvert" de type OneWire. *(Un fil)*

## Caractéristiques techniques du capteur DHT11 :

La mise en œuvre sur ARDUINO est d'autant plus facile que la bibliothèque DHT11.h est disponible en ligne. ***Elle se charge entièrement des protocoles de dialogue de type série.***

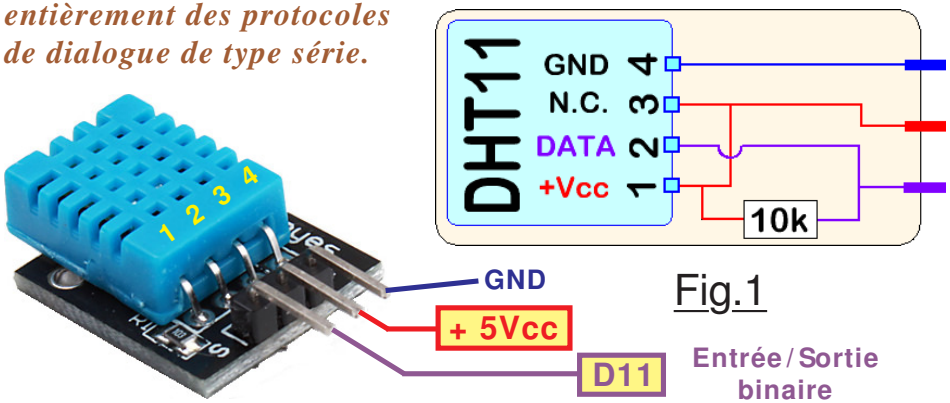



Fig.1

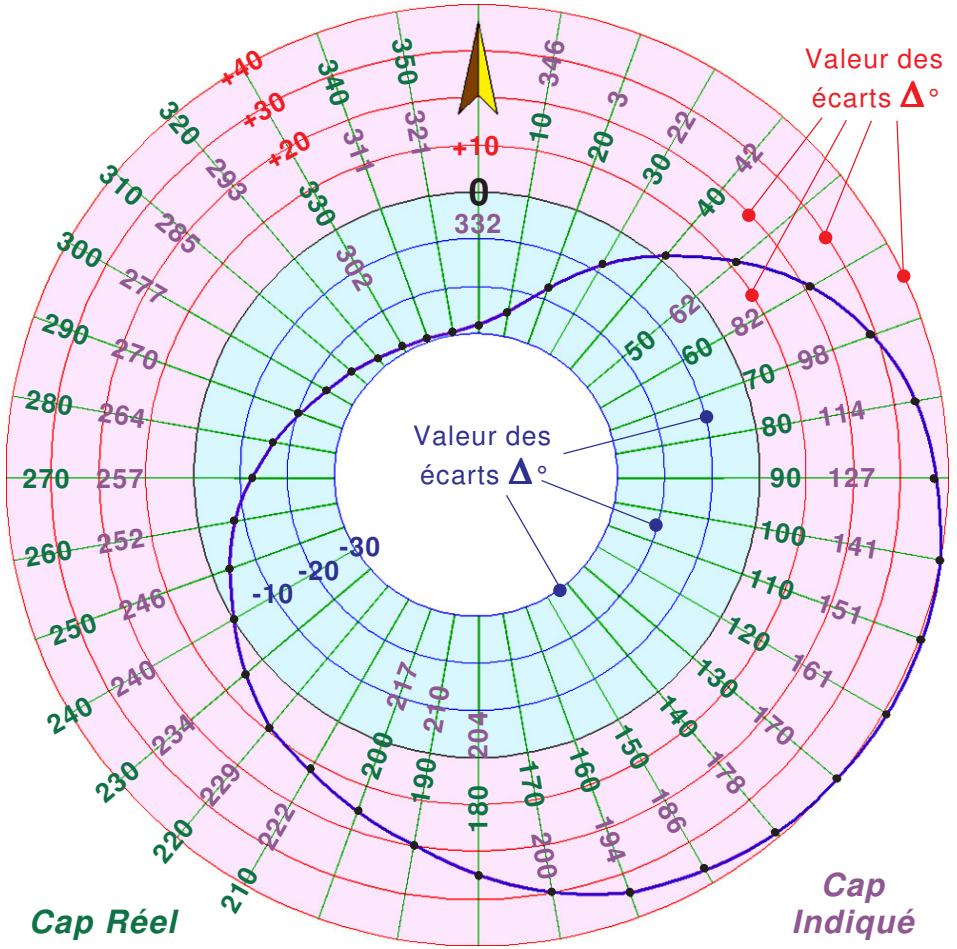
- Mesure de l'humidité relative. *(RH)*
- Alimentation comprise entre 3 Vcc et 5.5 Vcc.
- Courant d'alimentation : 100 µA maximum en veille, 2.5 mA maximum en dialogue.
- Mesures des températures comprises entre 0 °C et 50 °C.
- Mesures d'humidité entre 30 % et 90% RH à 0°C.
  entre 20 % et 90% RH à 25°C.
  entre 20 % et 80% RH à 50°C.
- Résolution 1 % RH. • Précision à 25 °C : ± 5% RH.

Lorsque le P.C. envoie un signal de démarrage, le DHT11 passe du mode faible consommation électrique au fonctionnement en mode dialogue et envoie l'accusé de réception suivi des données. *(Si le signal de départ est valide)* Il repasse ensuite en mode veille jusqu'à la prochaine sollicitation.

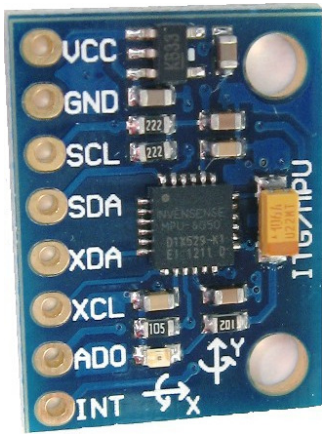
Fiche n°12

# Fiche de correction du compas magnétique.

| Réel | Indiqué | Réel | Indiqué | Réel | Indiqué | Réel | Indiqué |
|---|---|---|---|---|---|---|---|
| 0 | 332 | 90 | 127 | 180 | 204 | 270 | 257 |
| 10 | 346 | 100 | 140 | 190 | 210 | 280 | 264 |
| 20 | 3 | 110 | 151 | 200 | 217 | 290 | 270 |
| 30 | 22 | 120 | 161 | 210 | 222 | 300 | 277 |
| 40 | 42 | 130 | 170 | 220 | 229 | 310 | 285 |
| 50 | 62 | 140 | 178 | 230 | 234 | 320 | 293 |
| 60 | 82 | 150 | 186 | 240 | 240 | 330 | 302 |
| 70 | 98 | 160 | 194 | 250 | 246 | 340 | 311 |
| 80 | 114 | 170 | 200 | 260 | 252 | 350 | 321 |



# Centrale gyroscopique 3 axes MPU-6050. (1/4)




Nombreuses sont les applications qui doivent prendre en compte l'orientation d'un dispositif quelconque tel qu'une manette de jeu, un téléphone portable, un drone, un télescope, un caméscope gyro-stabilisé etc. Compte tenu de la complexité engendrée par la mesure des phénomènes d'accélérations tant linéaires qu'angulaires, un marché considérable à incité l'industrie à produire des composants de plus en plus sophistiqués, associés à des bibliothèques de programme pour en faciliter l'intégration dans les projets en cours de développement. Le MPU-6050 en est un exemple, qui comme bien d'autres circuits intégrés, à poussé les fournisseurs à mettre sur le marché des petits circuits imprimés tel que celui de la photographie donnée ci-dessus associant au composant de base quelques éléments électroniques pour en faire un dispositif prêt à l'emploi. Le circuit intégré basé sur un MPU-6050 combine un accéléromètre 3 axes et un gyroscope 3 axes avec un circuit DMP™. *(Digital Motion Processor™)* Le circuit de traitement DMP™ permet de résoudre les problèmes d'alignement sans rencontrer les erreurs générées par les composants discrets.

## Caractéristiques de base du circuit MPU-6050 :

- **Détecteur de mouvements intégrant 3 axes gyroscopiques et 3 axes accélérométriques linéaires et angulaires.**
- Alimentation DC : 2,3 à 3,4 Vcc.
- Consommation maximale : 3,9 mA avec les 6 capteurs activés.

**Accéléromètres :**

- Plages de mesure : ±2 g / ±4 g / ±8 g / ±16 g.
- Tolérance de calibration : ±3%.

**Gyroscopes :**

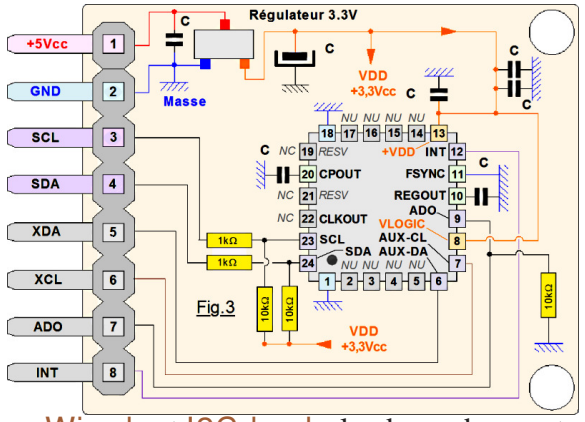
- Plages de mesure: ±250/500/1000/2000 °/s
- Tolérance de calibration : ±3%
- Interface de dialogue I2C fonctionnant jusqu'à 400kHz.
- Capteur de température intégré.
- Température de service : -40°C à +85°C

... / ...



# Centrale gyroscopique 3 axes MPU-6050. (2/4)

Le schéma électronique du petit circuit imprimé est donné ci-cntre. On peut y voir également le brochage du MPU-6050. Dans la version la plus simple de son utilisation, les quatre broches 1 à 4 sont suffisantes pour sa mise en œuvre. Si on développe le programme avec usage des bibliothèques Wire.h et I2Cdev.h, les branchements imposés sont ceux de la Fig.1 donnée ci-dessous.



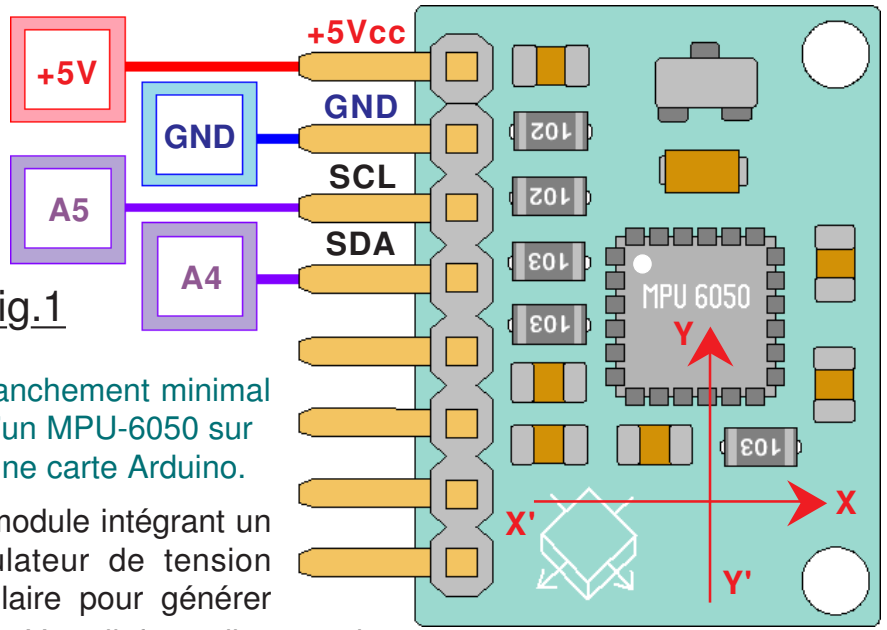




Fig.1

Branchement minimal d'un MPU-6050 sur une carte Arduino.

Le module intégrant un régulateur de tension tripolaire pour générer le 3.3Vcc, il faut alimenter le module en +5Vcc.

ATTENTION : L 'orientation des axes internes est celle montrée en rouge sur la Fig.1 et dépend de celle de l'implantation du 6050 sur le circuit imprimé. *(Ces axes sont fonction des développeurs.)*

... / ...

# Centrale gyroscopique 3 axes MPU-6050. (3/4)

## Principe de fonctionnement :

Le gyroscope / accéléromètre MPU-6050 comporte 6 axes avec une conversion analogique-numérique codée sur 16 bits simultanée sur chaque canal, et inclus une interface de dialogue I2C.
La lecture des mesures brutes de ce capteur est aisée, car il dispose d'un registre FIFO d'une taille de 1024 octets que le microcontrôleur d'Arduino peut lire par requête d'un signal d'interruption.
Le module fonctionne en esclave sur le bus I2C par rapport à Arduino *(Broches* ***SDA*** *et* ***SLC****.)* mais peut aussi contrôler un autre dispositif en aval avec les broches AUX-DA et AUX-CL. Le capteur avec son DMP est apte à réaliser des calculs rapides directement sur la puce à partir des mesures brutes du capteur, mais cette fonction est mal documentée, il est alors plus simple de traiter les mesures brutes avec une carte Arduino.

**Filtrage des mesures brutes :**

Un filtrage des mesures brutes s'impose si on ne veut pas cumuler des petites erreurs au cours du temps et laisser dériver le calcul de la position réelle. L'accéléromètre doit être filtré si on veut piloter un asservissement de manière fluide et sans oscillations permanentes avec surcompensations. On peut utiliser une simple moyenne glissante de 40 mesures de 12 ms chacune par exemple pour lisser les micro fluctuations ou utiliser un filtre de Kalman cette méthode étant nettement plus complexe.

## Nature des données habituellement utilisées :

- **Les quaternions** *(w, x, y, z)* **:** Trois quaternions permettent de définir l'orientation d'un axe, et le quatrième précise la valeur de la rotation autour de cet axe. *(Cette méthode contourne les limitations au delà de 180°.)*
- **Les angles d'Euler** *(Psi, téta, phi)* **:** Les angles d'Euler sont constitués par trois rotations autour des axes X, Y et Z.
- **Les angles en référence d'un mobile :** *(Voir la Fig.2)*
  *(C'est la forme d'informations généralement la plus commode.)*
  YAW : LACET. *( > 0 si la rotation est vers la droite.)*
  PITCH : TANGAGE. *( > 0 si la rotation est à piquer.)*
  ROLL : ROULIS. *( > 0 si l'inclinaison est vers la gauche.)*

... / ...

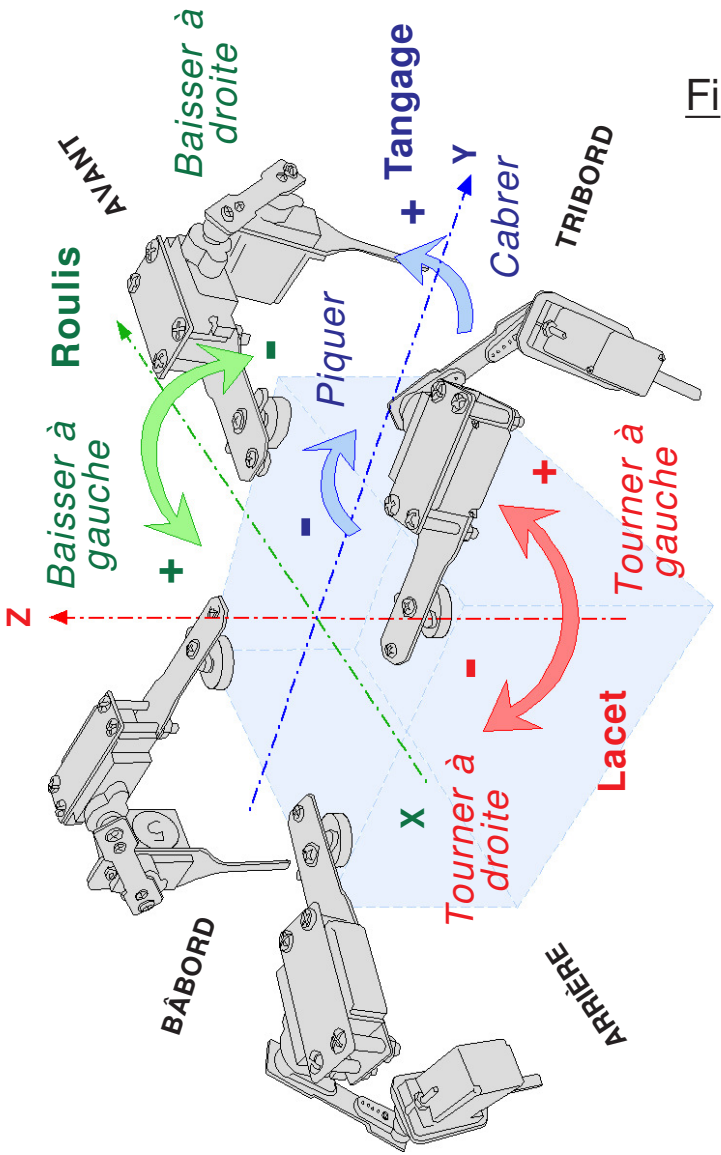


# Centrale gyroscopique 3 axes MPU-6050. (4/4)

Les sens de rotation correspondent à ceux d'un système trirectangle de sens direct. *(Sens triginométrique positif.)*

## Nom des axes gyroscopiques et sens de variation :

Fig.2




**Les angles en référence d'un mobile :**
YAW : LACET. *( > 0 si la rotation est vers la gauche.)*
PITCH : TANGAGE. *( > 0 si la rotation est à cabrer.)*
ROLL : ROULIS. *( > 0 si l'inclinaison est vers la gauche.)*

... / ...

## Liste des commandes sur un caractère.

a* : **A**llume les Phares OUI / NON.
b* : **B**loquer le niveau du LASER et des Phares.
c* : Bascule OUI/NON d'ACR de **C**onfiguration sur la ligne USB.
d* : **D**ébut du mode apprentissage. *(Si EEPROM effacée avec "y*".)*
e* : **E**nregistrer en EEPROM un balayage de télémétrie pour avoir la situation horizontale.
f* : **F**iger tous les moteurs.
g* : Retourne sur la ligne USB les composantes Gyroscopiques. *(Recale le gyroscope de Lacet si "=*" est activé.)*
h* : Active le balayage LASER en **H**orizontal. *(Bascule.)*
i* : **I**nformations sur l'état de la sonde. *(Bouclier, Sommeil, Rapide, Tension moteurs ...)*
j* : Jalonne : Retourne sur la ligne USB la distance Télémétrique.
k* : Fin du mode pilotage des moteurs en manuel. *(KILL.)*
l* : Active le **L**ASER. *(Bascule de type OUI / NON.)*
m* : Indique sur la ligne USB les valeurs **M**étéo. *(Luxmètre compris.)*
n* : Potentiomètre actif ou **N**eutralisé à 127.
o* : Retourne sur la ligne USB le cap de l'Orientation magnétique.
p* : Désactive la Torsion et replace en **P**osition au centre.
q* : **Q**uitter le programme : Passe en Configuration VEILLE et en SOMMEIL. *(Toutes les LED et le LASER sont éteints ...)*
r* : Mouvements coordonnés **R**apides. *(Bascule.)*
s* : Sommeil OUI / NON. *(LED et LASER éteints)*
t* : Active la correction de Lacet en **T**orsion.
u* : **U**tiliser un programme enregistré en mémoire non volatile.
v* : Active le balayage LASER en **V**ertical. *(Bascule.)*
w* : Restituer le balayage de télémétrie enregistré en EEPROM. (**W**aveform.)
x* : Lister sur la ligne USB le programme enregistré en EEPROM.
y* : Effacer un programme en EEPROM pour pouvoir débuter un autre apprentissage avec "d*".
z* : LAffiche la version du programme sur la ligne série.
=* : Arme le recalage du gyroscope de Lacet.
&* : Listage en continu des valeurs gyroscopiques. *(OUI / NON.)*
(* : Enregistrer un spectre colorimétrique en EEPROM.
)* : Afficher l'enregistrement chromatique disponible en EEPROM.

... / ...



## Liste des PROGRAMMES.

PGM Fonction réalisée.

p01* Posture Veille.
p02* Posture Stable Transversal.

| *Facteur de répétition* | pNr* | a | b | c | d | e |
|---|---|---|---|---|---|---|
| | Nb fois | 1 | 2 | 3 | 5 | 10 |

p03* Avancer d'un pas. *(Ou plusieurs si p1x*)*
p04* Reculer d'un pas. *(Ou plusieurs si p1x*)*
p05* Tourner à droite d'un pas. *(Ou plusieurs si p1x*)*
p06* Tourner à gauche d'un pas. *(Ou plusieurs si p1x*)*
p07* Décaler à droite d'un pas. *(Ou plusieurs si p1x*)*
p08* Décaler à gauche d'un pas. *(Ou plusieurs si p1x*)*

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| "p9a*" | Hanche | A | "p9b*" | Genou | A | "p9c*" | Pied | A |
| "p9d*" | Hanche | B | "p9e*" | Genou | B | "p9f*" | Pied | B |
| "p9g*" | Hanche | C | "p9h*" | Genou | C | "p9i*" | Pied | C |
| "p9j*" | Hanche | D | "p9k*" | Genou | D | "p9l*" | Pied | D |

p9m* Piloter un moteur au potentiomètre.
p10* Configuration Hauteur Maximale.
p11* Retour de configuration Hauteur Maximale.
p12* Configuration Stable LASER.
p13* Retour de Conf. Stable LASER vers Stable Transvrsl.
p14* Libérer les efforts.
p15* Tous les moteurs au neutre opérationnel.
p16* Configuration Stable Raisonnable.
p17* Retour de configuration stable raisonnable.
p18* Enregistrement en EEPROM de la posture actuelle.
p19* Utiliser l'enregistrement EEPROM de la posture.
p20* Lister l'enregistrement EEPROM de la posture.
p21* Dépose la sonde.
p22* Configuration atterrissage.
p23* Configuration décollage.

*En jaune les programmes qui acceptent la répétition avec "*" seule.*

| PGM | Cdx | PGM | Cdx | PGM | Cdx | PGM | Cdx | PGM | Cdx |
|---|---|---|---|---|---|---|---|---|---|
| p01* | p02* | p02* | p00* | p03* | p02* | p04* | p02* | p05* | p02* |
| p06* | p02* | p07* | p02* | p08* | p02* | p09* | p00* | p10* | p02* |
| p11* | p10* | p12* | p02* | p13* | p12* | p14* | p00* | p15* | p01* |
| p16* | p02* | p17* | p16* | p18* | p00* | p19* | p00* | p20* | p00* |
| p21* | p22* | p22* | p01* | p23* | p00* | | | | |

... / ...

## Protocoles d'utilisation de la sonde. (1/4)

**>>> Fin d'exploitation :** Toujours repasser en mode"q*". *(Quitter.)* Impératif pour le rangement à long terme de la sonde mais surtout pour la reprise car le programme sur RESET commence par configurer en mode VEILLE.

**La commande QUITTER "q*" :**

- Active les moteurs si ils sont figés et passe en option "Rapide",
- Pose la sonde sur le bouclier par un mouvement coordonné,
- Après une seconde passe en SOMMEIL et disjoncte @,
- Annule "Phares et LASER actifs" et "Torsion active",
- Neutralise le potentiomètre,
- Indique l'état par le message "Fin !" sur le moniteur de l'**IDE**.

**Reprise des activités après avoir QUITTÉ :**

- Vérifier la télémesure et l'état du calculateur avec "i*",
- Sortir du mode SOMMEIL : "s*", *(Trois LED s'allument.)*
- Réactiver la motorisation : "f*", *(La LED "Moteurs" s'éteint.)*
- Allumer les phares : *"a*", (Ils doivent éclairer même si le potentiomètre d'ajustement lumineux est au minimum.)*
- Centrer la commande potentiométrique,
- Réactiver le potentiomètre : "n*", *(La LED signalant l'option Potentiomètre neutralisé s'éteint.)*
- Armer le recalage gyroscope avec "=*", *(LED bleue allumée.)*
- Enregistrer le Cap actuel avec "g*", *(LED bleue éteinte.)*
- Vérifier intégralement l'état de la sonde avec "i*",
- Enchaîner avec les consignes d'exploitation désirées.

**La commande SOMMEIL "s*" :**

Cette fonction permet de faire passer la sonde en économie énergétique. La signalisation et les éclairages sont disjonctés. Avant de basculer la machine dans ce mode il est fortement recommandé de la configurer avec "p01*"en posture VEILLE. *(Posée sur le bouclier.)*
* L'option servomoteurs OFF est validée,
* Toutes les LEDs et le LASER sont éteints.

**Sortie du mode SOMMEIL :**

- Vérifier la télémesure et l'état du calculateur avec "i*",
- Sortir du mode SOMMEIL avec "s*",
- Reprendre les activités d'exploitation.

... / ...



## Protocoles d'utilisation de la sonde. (2/4)

**Enregistrement d'un programme en EEPROM :**

Tenter de commencer un apprentissage alors qu'un programme est déjà préservé en EEPROM génèrera une alerte "!ERR 9!". Si on veut effectivement changer de comportement automatisé il faut au préalable purger la mémoire par la commande "y*".
On peut parfaitement enregistrer quand les moteurs sont mis sur OFF avec "f*". L'état de la motorisation OFF n'est pas enregistré, donc on peut librement tester le programme, les mouvements étant figés pour sécurité. Quand la séquence mémorisée est validée, il est alors possible de réactiver les servomoteurs et déclencher la séquence automatique apprise. Durant le mode enregistrement la LED rouge dédiée "Enregistrement" est allumée.

**Toute commande sur un caractère fait sortir du mode** apprentissage donc seules les instructions genre programme de type "pNN*" seront mémorisables. Le nombre maximum d'instructions enregistrées est de 30.

- Éventuellement motorisation mise sur OFF avec "f*",
- Mémoire purgée débuter l'apprentissage avec "d*", La LED rouge d'apprentissage s'illumine. La sonde informe du mode par le message d'état "APPRENTISSAGE",
- Chaque instruction enregistrée ajoute une information d'ordre de type "N/30" en début d'accusé de réception.

Toute nouvelle instruction en mode apprentissage déjà saturé à 30 consignes engendrera une alerte "!ERR 11".

**Utilisation d'un programme disponible en EEPROM :**

Si la mémoire d'apprentissage est effacée : alerte "!ERR 8".

- Figer la motorisation avec "f*",
- Déclencher la séquence automatique avec "u*" pour en vérifier la cohérence et le bienfondé. *(Elle peut contenir des alertes suite à des préalables non effectifs etc.)* Chaque commande reproduite informe de son gang par des messages de type ">>> pNN*",
- Séquence apprise correcte valider la motorisation avec "f*",
- Faire réaliser la séquence mémorisée en EEPROM en réitérant la commande de déclenchement "u*". *(Suite en Fiche 3/4.)*

... / ...

## *Protocoles d'utilisation de la sonde.* (3/4)

**Effacer un programme de déplacements en EEPROM :**

- Allumer les phares avec "a*", *(Première sécurité.)*
- Imposer le retour de configuration sur la ligne USB avec la commande "c*", *(Deuxième sécurité.)*
- Effacer le programme actuel en EEPROM avec "y*".

Si la double sécurité Phares allumés et ACR avec configuration actuelle ne sont pas effectifs il se produira une "!ERR 10". *(Consigner un effacement alors que la mémoire est purgée ne provoque pas d'alerte, on "surcharge" des $FF en EEPROM.)*

**Lister un programme d'apprentissage présent en EEPROM :**

La commande "x*" engendre le listage sur la ligne série du moniteur de l'**IDE** de toutes les instructions de la séquence actuellement en mémoire non volatile EEPROM. Si cette dernière est purgée il y a génération d'une alerte "!ERR 8!".

**Configuration mémorisée en EEPROM :**

Trois instructions qui entrent dans le cadre d'un apprentissage permettent de sauvegarder et d'utiliser une posture particulière enregistrée en mémoire non volatile EEPROM. À tout moment on peut enregistrer la configuration actuelle, et en particulier lorsqu'une attitude spécifique a été réalisée en mode pilotage manuel.

- "p18*" **réalise l'enregistrement en EEPROM de la posture actuelle** si la double sécurité est effective. *(Phares allumés avec "a*" et accusés de réception informant de la configuration actuelle par la commande "c*".)* ! ERR 20! si la double sécurité n'est pas satisfaite.
- "p19*" fait passer la sonde en posture mémorisée.*(S'il n'y a pas une alerte de tension insuffisante sur la motorisation.)*
- "p20*" liste sur le moniteur de l'**IDE** la configuration mémorisée.

**Résumé des erreurs pour le mode apprentissage :**

! ERR 8! : Vouloir lister avec "x*" alors que la mémoire est vide.
! ERR 9! : Effacer avec "y*" alors qu'un programme est présent.
! ERR 10! : Effacer avec "y*" sans la double sécurité effective.
! ERR 11! : Mémoire d'apprentissage saturée à 30 instructions.
! ERR 20! : Tentative d'utiliser "p18*" sans la double sécurité.

... / ...



## *Protocoles d'utilisation de la sonde.* (4/4)

**Pilotage manuel des moteurs par le codeur rotatif :**

- Bloquer les éclairages avec "b*" qui allume la LED blanche,
- Début > Figer la motorisation avec la commande "f*",
- Centrer le bouton du codeur rotatif,
- Si nécessaire stopper le mode Capteur Neutralisé avec "n*",
- Passer en mode manuel avec "p9m*", la lettre 'm' précise le moteur piloté comme indiqué sur la Fiche n°12. L'accusé de réception est complété avec une information de type "Genou B" si 'm' était égal à 'e' par exemple, précisant l'articulation et la Jambe concernées. La LED rouge dédiée éclaire informant visuellement du mode pilotage manuel.
- Activer la motorisation avec la commande "f*",
- Avec le codeur modifier à convenance l'orientation du moteur,
- Éventuellement valider "c*" et noter la valeur de consigne.
- Pour changer de moteur invalider "c*" et reprendre en Début >,
- Quand la configuration désirée est obtenue, "k*" pour terminer le mode manuel. *(KILL.)* La LED rouge dédiée s'éteint,
- Si désiré, sauvegarder la posture avec : "P18*",
- Éventuellement invalider l'option "b*".

Durant l'intégralité de cette procédure on peut utiliser les commandes à un caractère, et surtout "i*" qui précise l'état de la sonde.

**Piloter le LASER :**

- La sonde est en posture ***Stable Transversal***, *("P02*".)*
- Passer en posture ***Stable LASER*** avec "P12*",
- Allumer le LASER avec "l*",
- Centrer le codeur rotatif,
- "v*" ou "h*" à convenance, *(La LED de V ou H s'allume.)*
- Orienter en vertical ou en horizontal avec le codeur rotatif.

**Sortie d'une procédure LASER :**

- Imposer l'orientation Horizontale avec "h*",
- Amener visuellement la Hanche A en orientation proche de la posture pour ***Stable Transversal***,
- Vérifier avec "i*" que Conversion CAN fait approximativement ≈ 240 (**≈80 à ≈100**),
- Éteindre le LASER avec "l*", *(La LED de V ou H s'éteint.)*
- Revenir de la posture Stable LASER avec "P13*".

... / ...

## *Liste des ERREURS en exploitation.*

| ERR | Nature de l'erreur de syntaxe. |
|---|---|
| 1 | Le message ne commence pas par "p". |
| 2 | Message non terminé par "*". |
| 3 | Une consigne sur trois caractères est invalide. |
| 4 | Consigne à plus de quatre caractères invalide. |
| 5 | Lettre de répétition de mouvement incorrecte. |
| 6 | Le caractère "C*" n'est pas valide. |
| 7 | "pNN*" avec 00 ou supérieur au nombre possible. |
| 8 | Pas de programme enregistré en mémoire EEPROM. |
| 9 | PGM présent en EEPROM. *(L'effacer avec "y*")* |
| 10 | Tentative d'effacer un programme en EEPROM alors que la double sécurité "Phares allumés" et ACR validé n'est pas satisfaite. |
| 11 | Saturation de la mémoire d'enregistrement. |
| 12 | Bouclier non posé sur le sol martien. *(Ou sur le berceau dans le lanceur Ariane.)* |
| 13 | "h*" ou "v*" alors que le LASER n'est pas actif. |
| 14 | "h*" ou "v*" sonde non "Configurée LASER". |
| 15 | "p03*" avec un obstacle à moins de 8cm. |
| 16 | Capteur Humidité : Temps d'attente dépassé. |
| 17 | Capteur Humidité : Erreur de "Checksum". |
| 18 | "p9c*" : Caractère c pour désigner le moteur incorrect. |
| 19 | "p09*" : Pas de caractère c pour désigner le moteur. |
| 20 | Tentative d'enregistrer une posture en EEPROM alors que la double sécurité "Phares allumés" et ACR validé n'est pas satisfaite. |
| 21 | Invoquer un programme alors que le programme préalable n'est pas correct. *(Voir la liste des conditions préalables sur la fiche : Liste des PROGRAMMES.)* |
| 22 | Tension motorisation inférieure à 4.0Vcc. |
| ---------- | Problèmes sur la centrale gravitationnelle ---------- |
| 30 | Échec de connecxion avec le MPU6050. |
| 32 | Débordement FIFO en dialogue "Inertiel". |
| 33 | Précédé de 1 : Échec de chargement initial.<br>Précédé de 2 : Échec MAJ Config du MPU. |



## *Tests de validation de la carte Arduino NANO.*

01) RESET : 1 !ERR 33! suivi de !ERR 30! suivi de >>>,
02) "**z***" : Version P30-T5. suivi de ! PGM z OK!,
03) "**abcd**" puis valider : Il ne se passe rien,
04) "**e***" : abcde* > !ERR 7!, *(C'est "*" qui déclenche la prise en compte du texte frappé sur le moniteur. Ici une erreur.)*
05) "**02m***" : 02m*> !ERR 1!,
06) "**p2***" : p2*> !ERR 3!,
07) "**p2c***" : Repete 3 suivi de ! PGM 2 OK!,
08) "**p2f***" : p2f*> !ERR 5! suivi de ! PGM 2 OK!,
09) "**C***" : C* > !ERR 6!, *(Attention : Caractère "C" majuscule.)*
10) "**y***" : y* > !ERR 10!
11) "**a***" : a* > ! PGM a OK!, *(Prépare la commande "y*".)*
12) "**y***" : y* > !ERR 10!,
13) "**c***" : c* > ! PGM c OK! *(Prépare la commande "y*".)*
Posture = 239 213 307 344 414 314 222 ... 388 314,
14) "**y***" : y* > ! PGM y OK!, *(Accepte l'effacement du PGM.)*
15) "**c***" : c* > ! PGM c OK! *(N'affiche plus la posture actuelle.)*
16) "**d***" : d* > suivi de APPRENTISSAGE, et de ! PGM d OK!,
17) "**p02***" : p02*> suivi de 1/30 ! PGM 2 OK!,
18) "**p03***" : p03*> suivi de 2/30 !ERR 15! et de Distance : 0cm,
19) "**p4c***" : p4c*> suivi de 3/30, de Repete 3 et de ! PGM 4 OK!,
20) "**x***" : x*> suivi de p02* / p03* / p4c* puis ! PGM * OK!,
21) "**l***" : l* > ! PGM l OK! *(Allume le LASER.)*
22) "**n***" : n* > ! PGM n OK! *(Potentiomètre forcé à 127.)*
23) "**b***" : b* > ! PGM b OK! *(Bloquer phare et LASER.)*
24) "**r***" : r* > ! PGM r OK! *(Moteurs Rapides.)*
25) "**i***" : i* > ----------

***Liste les paramètres de configuration de la sonde. LASER et phares sont allumés à 127, Sommeil NON etc,***

26) "**&***" : Liste en continu les valeurs gyroscopiques,
27) "**&***" : Stoppe le listage en continu, ⇦ Attendre le OK.
28) "**(***" : (* > ! PGM ( OK! *(Enregistre un spectre couleur.)*
29) "**)***" : )* > Puis liste à l'écran les valeurs enregistrées,
30) "**u***" : u* > suivi de Repete PGM ... FIN, ⇦ Attendre FIN.
31) "**q***" : q* > FIN ! suivi de ! PGM q OK! *(Toujours finir ainsi.)*

***Si toutes ces instructions fonctionnent, c'est que l'ATmega328 est correctement programmé et bon pour le service.***

# Modifier une TABLE de POSTURE.

À titre d'exemple nous allons dans cette fiche affiner l'orientation de la Hanche de la Jambe A pour la posture de décollage, car avec la présence du LASER le risque d'interférence devient patent. ***Procédure de modification de la posture :***

1) Positionner le bouton du potentiomètre au neutre. *(Centré.)*
2) Mettre en service la sonde. *(Dans notre cas elle doit reposer sur le berceau et le micro contacteur activé.)*
3) Valider l'affichage de la configuration : "c*".
4) Activer le programme de rétraction avec "p23*". Confirmer la sonde placée sur le berceau avec "o*".

ATTENTION : La posture va amener le LASER de Jambe A en légère butée sur la Jambe B. Il importe de rapidement supprimer cette contrainte, ou les servomoteurs risquent de surchauffer :

5) Débloquer la hanche avec "p9a*".
6) Débloquer Neutre du potentiomètre "n*".
7) Avec le potentiomètre décaler la Hanche.
8) Commande "i*" pour faire afficher la nouvelle configuration.

```
Posture = 239 309 419 344 294 186 222 295 423 346 284 194
p23* >
Sonde sur berceau ?
!PGM 23 OK!
Posture = 200 152 315 413 486 316 159 139 289 411 463 325
i* >
------------------------------
Bouclier au sol : OUI
!PGM i OK!
Posture = 211 152 315 413 486 316 159 139 255 411 463 325
```

9) Dans **P29** modifier la ou les valeurs de la posture et téléverser le programme pour écrire les valeurs en EEPROM.

```
//------------------ Tableau de Configuration Décollage. ----------
Ecri ROM(PTR, 211); Ecri PROM(PTR, 152); Ec M(PTR, 315); // Jambe A.
Ecri ROM(PTR, 413); Ecri PROM(PTR, 486); Ec M(PTR, 316); // Jambe B.
Ecri ROM(PTR, 159); Ecri PROM(PTR, 139); Ec M(PTR, 255); // Jambe C.
Ecri ROM(PTR, 411); Ecri PROM(PTR, 463); Ec M(PTR, 325); // Jambe D.
```

10) Téléverser à nouveau **P30** et vérifier la posture modifiée.



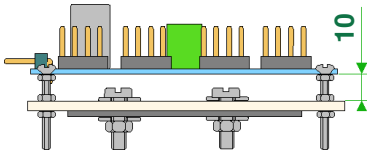
# Notes personnelles.

# ASSEMBLAGE et INTÉGRATION. 1/2

1) Placer le support de la boussole statique sur la plaque qui supporte le multiplexeur. Engager le connecteur sur le support HE14 mâle.
2) Approcher le circuit imprimé du support du condensateur de 470µF et visser le + et le - sur le connecteur vert du multiplexeur.

*Pour le moment de circuit imprimé du condensateur de 470µF n'est pas encore immobilisé car il faut pouvoir insérer dessous les vis ϕ M3 qui immobilisent la plaque support sur le châssis.*

3) Le module boussole HMC6352 n'étant pas sur son connecteur, assembler le module multiplexeur sur sa plaque support. *(Écart environ 10mm.)*



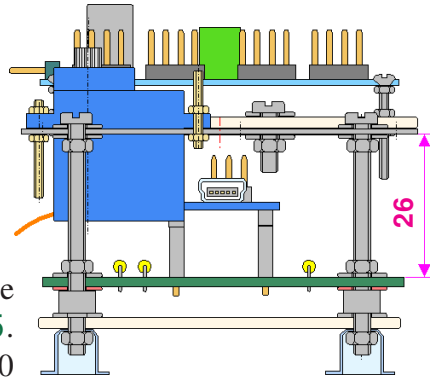

4) Assembler la plaque support du module multiplexeur sur le châssis à l'aide des quatre boulons ϕ M3.
5) Assembler le circuit imprimé du condensateur de 470µF sur la plaque support du multiplexeur.
6) Placer la protection des faisceaux de fils entre les lignes HE14 qui supportent Arduino NANO et enficher ce dernier sur son support.
7) Placer le masque en carton des LED de la carte Arduino NANO.
8) Préparer le circuit imprimé principal en disposant conformément à Image 36.JPG les lignes qui vont vers le multiplexeur pour sa commande sur le coté et sur le dessus pour S12 à S15.
9) Positionner les écrous sur les vis longues de structure pour que le circuit imprimé principal soit à 26mm du châssis en aluminium. Enfiler les rondelles d'appui métalliques qui seront sur le dessus su C.I. *(Aucun capteur n'est en place ni les "straps" à languette.)*
11) Introduire le circuit imprimé principal sur les vis longues de structure. *(Attention : Les torons des deux faisceaux ont tendance à se coincer avec les rondelles métalliques.)*
12) Pousser le circuit imprimé vers l'avant.
13) Introduire les rondelles isolantes, les entretoises et les rondelles métalliques. Glisser provisoirement le bouclier pour pouvoir poser la sonde sur un berceau.
14) Brancher les deux connecteurs HE14 sur le multiplexeur.
15) Brancher la ligne d'alimentation la ramener vers l'avant. Si option autonome envisagée, brancher le cordon ombilical.

... / ...

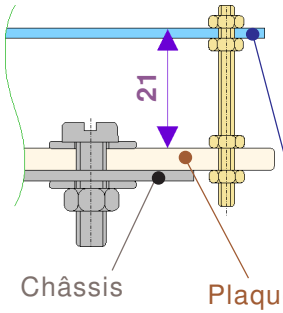
Fiche n° 20

# ASSEMBLAGE et INTÉGRATION. 2/2

16) Placer les écrous dans les "sabots", enlever le bouclier, et faire passer les deux lignes d'exploitation sous le circuit imprimé principal.
17) Réintroduire le bouclier.
18) Positionner la protection des câbles en sortie des deux lignes électriques d'exploitation.
19) Introduire les écrous et les sabots et les serrer modérément.
20) Brancher les lignes des divers servomoteurs sur le module multiplexeur ainsi que S12 à S15.
21) Brancher le module MPU-6050 sur le circuit imprimé qui supporte la cellule photorésistante




22) Installer le circuit imprimé qui supporte la cellule photorésistante dont le dessus se trouve à 21mm du dessus de la plaque qui supporte le module multiplexeur.
23) Mettre en place le petit module de la boussole statique HMC6352.
24) Placer sur les deux vis support le filtre du spectrographe colorimétrique.




## ➢ Dépose du circuit imprimé principal.

01) Enlever le télémètre à ultrasons et le module humidistance.
02) Débrancher la centrale gyroscopique.
03) Déconnecter la ligne de puissance des servomoteurs.
04) Libérer la ligne de la cellule photorésistante.
05) Débrancher le conducteur qui va au LASER.
06) Déconnecter du C.I. principal la ligne I2C à 5 broches.
07) Libérer du C.I. principal le 6 broches HE14 du cordon ombilical.
08) Débrancher la prise à 4 fils qui pilote le multiplexeur.
09) Libérer le connecteur HE14 de S12 à S15 sur le multiplexeur.
10) Déposer le bouclier puis le circuit imprimé principal.

... / ...

# *Petite alimentation 5Vcc - 1.*

Prévu pour fonctionner sous 5Vcc le petit ventilateur V consomme 65mA. Possibilité d'augmenter la tension en sortie avec le potentiomètre ajustable jusqu'à 7Vcc. **Dans ce cas couper le ventilateur**.

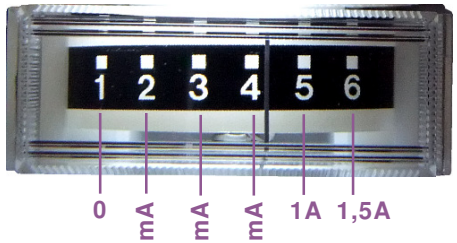
| $U_{sortie}$ | $I_{MAX}$ | $I_{indiqué}$ |
|---|---|---|
| 6v | 1A | 5,7v |
| 7v | 700mA | 6,8v |

**Bouton de Panique BP :** Sur incident BP annule immédiatement la tension en sortie mais laisse tourner le ventilateur. La LED D2 s'éteint. ***Débrancher la sortie ou le bloc secteur avant de relâcher le bouton poussoir*** qui au repos rétablit le contact électrique. *(Voir la Fig.1)*

## ➢ *Comportement du galvanomètre.*

Compact et rétroéclairé, il présente toutefois l'inconvénient de commencer les graduations non pas à zéro mais à 1, compliquant de ce fait l'interprétation des affichages. La Fig.2 présente l'étalonage, le graphe de la Fig.3 précisant qu'à partir de 4 il faut ventiler.

Fig.2



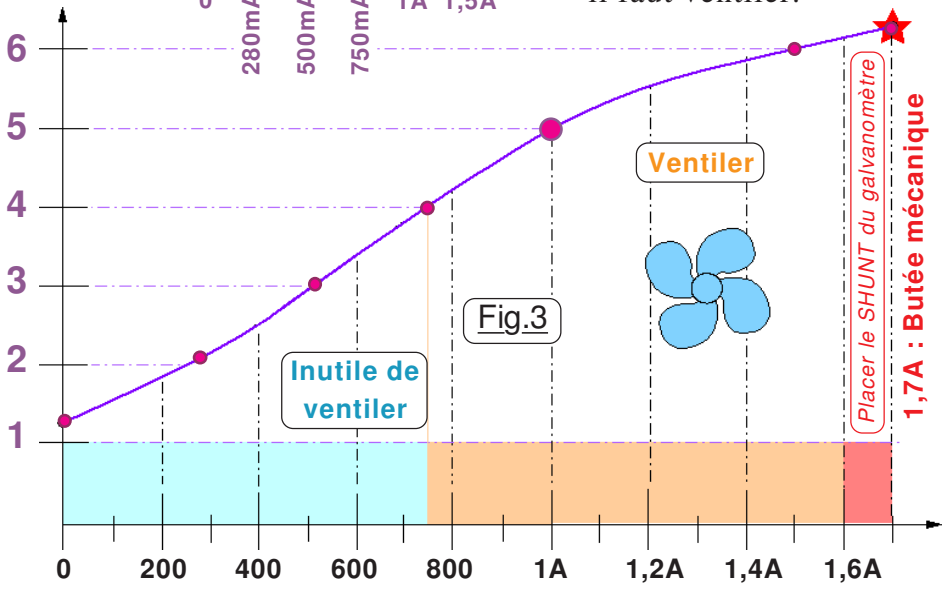
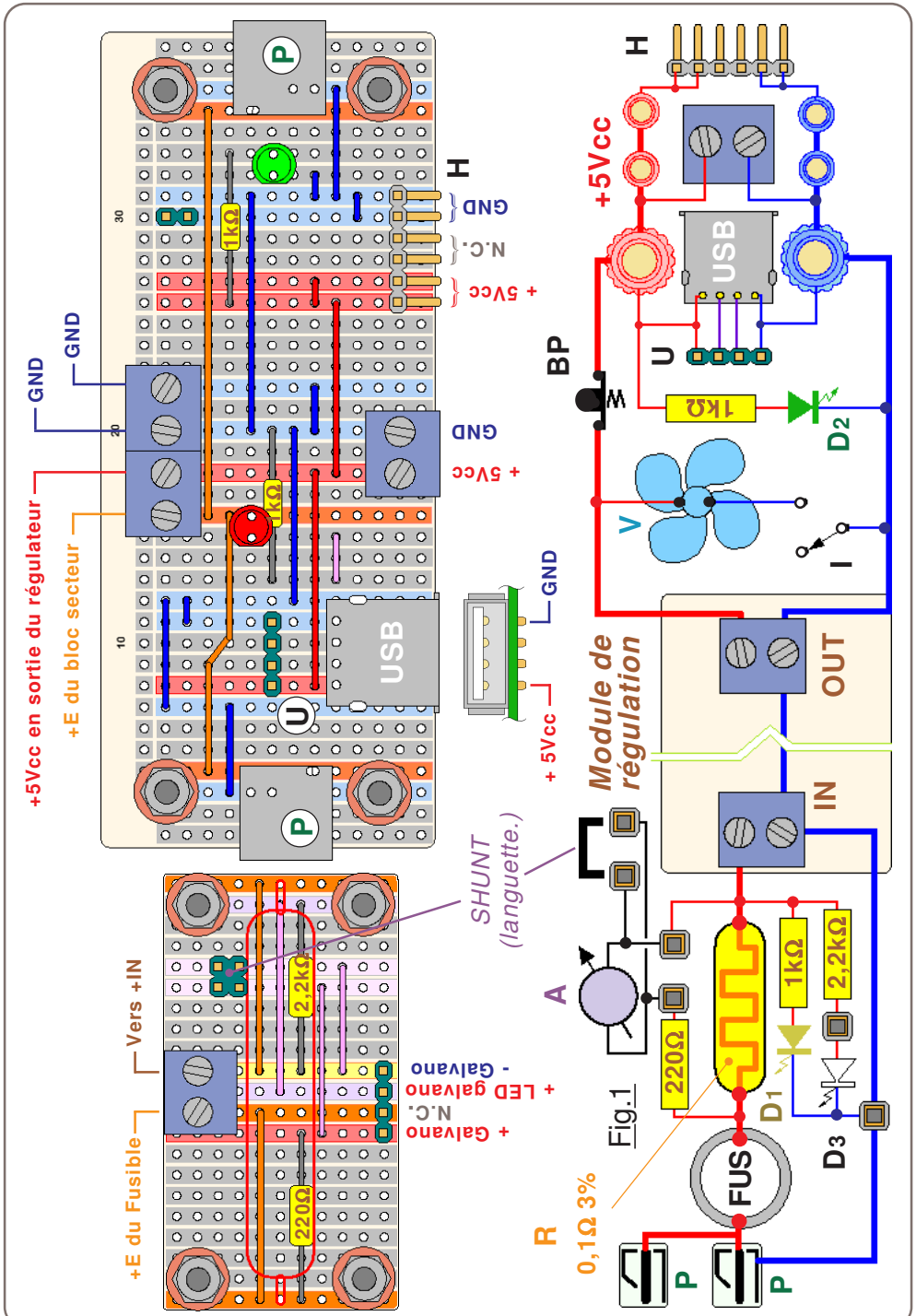




Fig.3

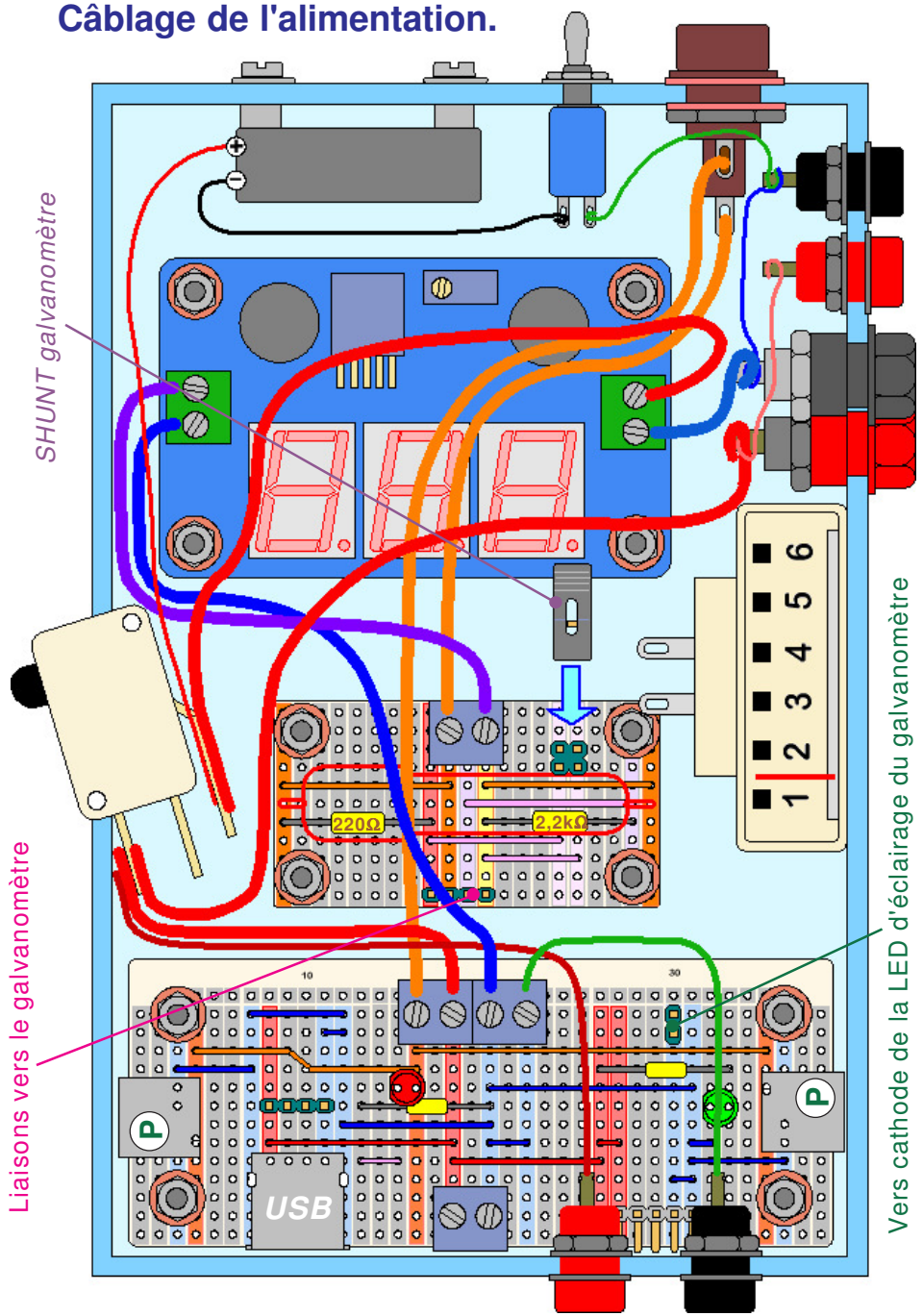
... / ...






... / ...

## Câblage de l'alimentation.




… / …



# *Petite alimentation 5Vcc - 2.*

### ➢ *Comportement du module régulateur de tension.*

Jusqu'à un débit de 750mA il n'est pas utile de ventiler. Dès que l'indication d'intensité dépasse la graduation **4** il importe de mettre en fonctionnement le petit ventilateur. Les indications de tension d'entrée et de sortie sur l'afficheur ne sont pas très précises. Le tableau donné ci-dessous précise les valeurs réelles en fonction de celles affichées ainsi que les conditions de fonctionnement.

| V IN | Aff. | I | V OUT | Aff. |
|---|---|---|---|---|
| 11V | 11,2 | à vide | 5,05v | 4.8 |
| 10,2V | 9,7 à 10,2 | 500mA | 5v | 4.8 |
| 10V | 9,9 à 10,3 | 400mA | 5v | 4.8 |
| 9V | 9,2 à 9,7 | 700mA | 5v | 4.7 |
| 9,6V | 8,9 à 9,7 | 750mA | 4.98v | 4.8 |
| 9V | 8,2 à 9,1 | 1A | 4.85v | 4.8 |
| 7,95V | 6,5 à 7,7 | 1,5A | 4.7v | 4.4 |
| 7,2V | 6,2 à 7,7 | 1,7A | 4.5v | 4.5 |

On observe que la tension affichée en entrés du régulateur fluctue rapidement. C'est assez normal car le bloc basse tension secteur présente une filtration sommaire. Les oscillogrammes de la Fig.4 ont été saisis en mode alternatif. Celui montré en **A** présente un RAC résiduel de 0,9V crête à crête environ pour une intensité consommée sur **OUT** de 750mA. On constate en **B** que pour une intensité débitée de 1A la composante alternative avoisine 2,5V crête à crête. Sur le tracé **C** la tension alternative arrive à 3V crête à crête pour une intensité débitée de 1,5A sur **OUT**. La marge de tension différentielle entre la sortie et l'entrée du régulateur commence à devenir faible quand la tension redressée passe par sa valeur minimale.

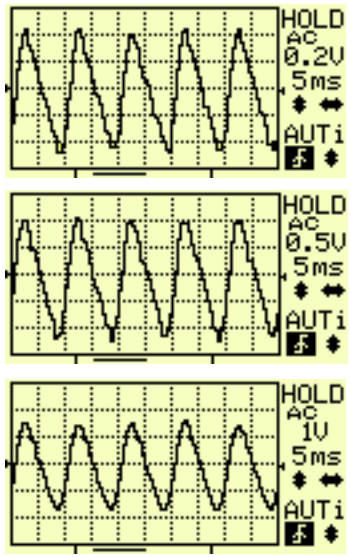



Fig.4

… / …

## *Petite alimentation 5Vcc - 3.*

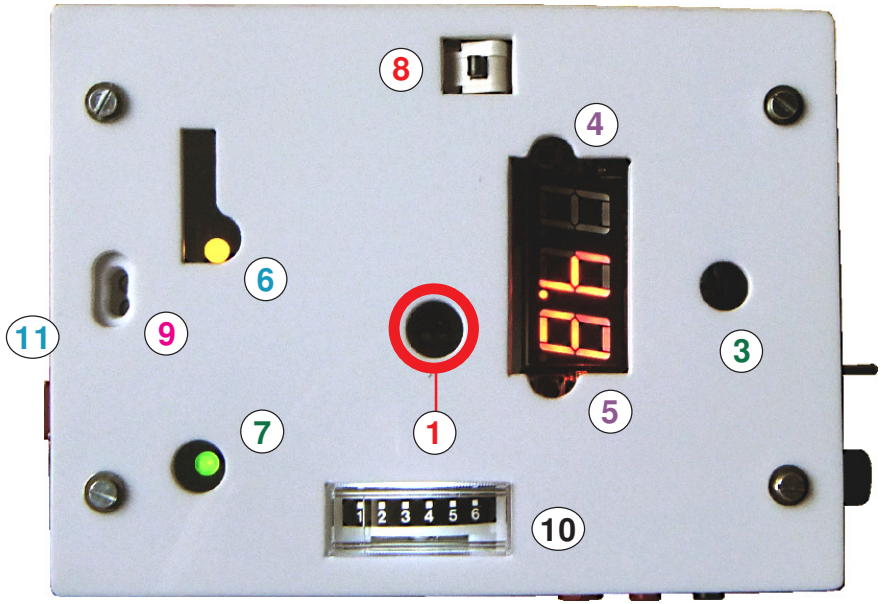



Fig.1

➢ ***Mise en œuvre.***

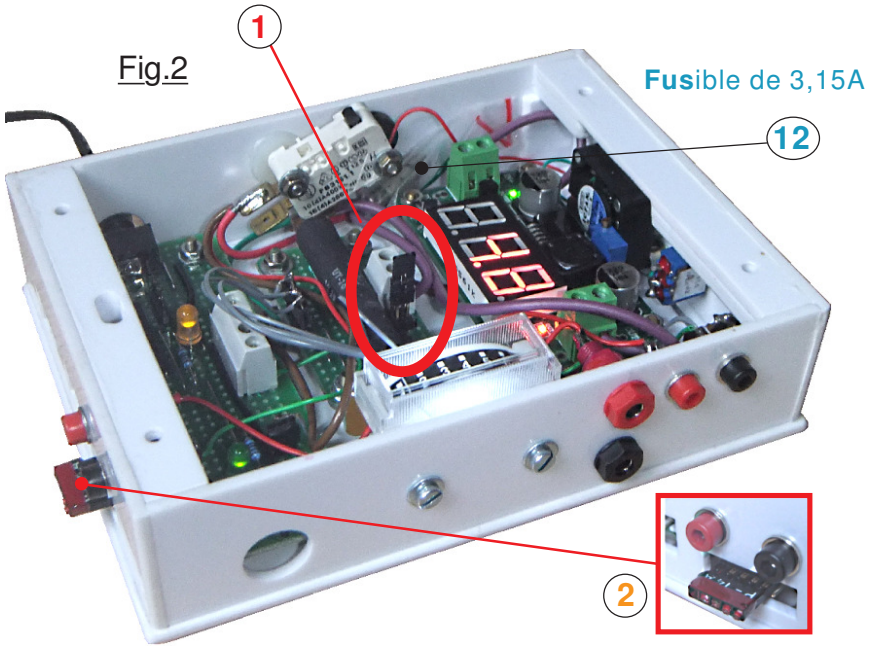
Il est possible de faire débiter cette alimentation à 1,7A en permanence à condition d'activer impérativement le petit ventilateur. Le galvanomètre **10** est alors en surcharge. Sa déviation exagérée sur le long terme pourrait en diminuer la fiabilité. Par ailleurs, certains moteurs peuvent engendrer des pics d'intensité à leur démarrage qui souquent énergiquement le cadre mobile, brutalité qui ne va pas dans le sens de la longévité. Pour parer ces deux cas de figure relativement peu fréquents, la possibilité de placer **10** en court-circuit est prévue sur le circuit imprimé sous la forme d'un connecteur HE14 double et de grande hauteur. Ainsi, par le truchement d'une petite languette en **1** il devient facile d'établir le contact ou au contraire de le supprimer. Le connecteur HE14 est utilisé pour avoir les broches proches du couvercle et ainsi insérer facilement la languette de court-circuit coffret fermé.
Autre risque potentiel : Les broches du petit connecteur HE14 de sortie sur le coté droit en **11** dépassent légèrement du coffret. De ce fait un risque de contact accidentel avec l'environnement ou d'un

... / ...

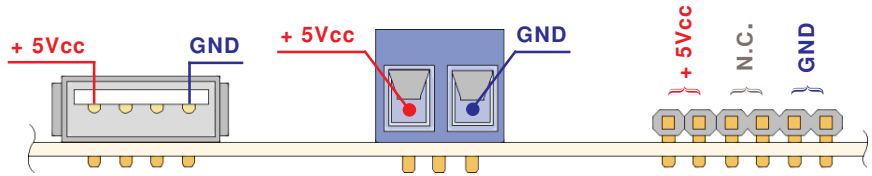


court-circuit malencontreux devient trop important si elles ne sont pas utilisées. En **2** un bouchon isolant pare ce risque.

Fig.2




En **3** on peut ajuster la tension de sortie avec un petit tournevis idoine. Un quelconque stylet permet d'allumer ou d'éteindre l'afficheur en **4** et de choisir la valeur de la tension d'entrée ou de sortie indiquée en **5**. En **6** la lumière est assez grande pour permettre la sortie des fils qui seraient reliés au connecteur USB et pour voir la LED témoin de la présence de tension en entrée du régulateur. L'orifice **7** laisse voir la LED témoin de la tension en sortie qui s'éteint quand on active en **8** le "bouton de panique". Enfin en **9** il sera facile avec un petit tournevis de brancher des fils de section suffisante sur le bornier de sortie disponible sur le coté gauche.
Pas très visible sur la Fig.2 on peut observer en **12** un petit sachet en matière thermoplastique transparente dans lequel est disponible un fusible de rechange calibré en nominal à 3,15A.




... / ...