Hes·so VALAIS WALLIS
Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

Filière Systèmes industriels

Orientation Power & Control

Diplôme 2015

- Non confidentiel -

Vincent Giachino

Bioluminescence

Professeur
Prof. Martial Geiser

Expert
Prof. Jan Roelof Van Der Meer

Date de la remise du rapport
10.07.2015

HES-SO Valais

SI	TV
X	X

Données du travail de diplôme
Daten der Diplomarbeit

FO 1.2.02.07.BB
haj/31/03/2014

☒ FSI ☐ FTV	Année académique / *Studienjahr* **2014/15**	No TD / *Nr. DA* **pc/2015/32**
Mandant / *Auftraggeber* ☒ HES—SO Valais ☐ Industrie ☐ Etablissement partenaire *Partnerinstitution*	Etudiant / *Student* **Vincent Giachino** Professeur / *Dozent* **Martial Geiser**	Lieu d'exécution / *Ausführungsort* ☒ HES—SO Valais ☐ Industrie ☒ Etablissement partenaire *Partnerinstitution*
Travail confidentiel / *vertrauliche Arbeit* ☐ oui / ja [1] ☒ non / nein	Expert / *Experte* *(données complètes)* **Prof. Jan van der Meer, Department of Fundamental Microbiology, UNIL, 1015 Lausanne (JanRoelof.VanDerMeer@unil.ch)**	

Titre / *Titel*

Bioluminescence

Description / *Beschreibung*

BRAAVOO est un projet européen qui vise à développer des solutions innovantes pour la mesure en temps réel et in situ d'impact élevé et difficile à mesurer de polluants marins. Le concept de BRAAVOO repose sur une combinaison unique de trois types de biocapteurs, qui permettra simultanément la détection d'un certain nombre de polluants marins spécifiques et prioritaires marins ainsi que des polluants globaux dont les effets biologiques généraux peuvent servir d'alerte rapide. Des bouées avec une autonomie d'au moins un mois seront construites. Ce projet est coordonné par l'UNIL qui a développé avec ses partenaires un chip contenant plusieurs biosenseurs basés sur des bactéries modifiées génétiquement. Ces dernières deviennent bioluminescentes en présence d'un polluant.
Dans le cadre de ce projet, l'UNIL va intégrer ses bactéries dans des chips, contenant chacun 10 puits, construits par une entreprise. Ces chips seront ensuite montés dans un système de mesures automatiques qui va quotidiennement mesurer des prélèvements d'eau. Ce système devra injecter l'eau à mesurer, effectuer séquentiellement les mesures de bioluminescence de chaque puit d'un chip et passer au chip suivant.

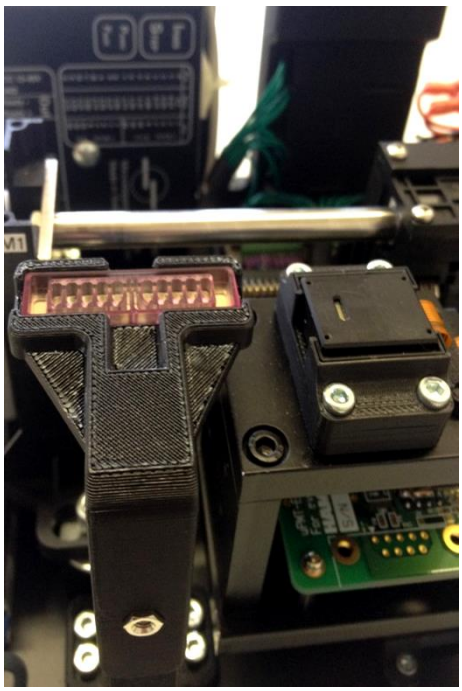
Objectifs / *Ziele*

— Fermer hermétiquement les chips (UNIL)
— Développer et construire la partie qui mesure séquentiellement la bioluminescence des puits d'un chip durant deux heures.
— Analyser les résultats et calculer les concentrations des polluants mesurés.

Signature ou visa / *Unterschrift oder Visum* Responsable de l'orientation *Leiter der Vertiefungsrichtung:*..... [1] Etudiant / Student :	Délais / *Termine* Attribution du thème / *Ausgabe des Auftrags:* 11.05.2015 Remise du rapport / *Abgabe des Schlussberichts:* 10.07.2015, 12:00 Expositions / *Ausstellungen der Diplomarbeiten:* 26 – 28.08.2015 Défense orale / *Mündliche Verfechtung:* Semaine | Woche 36

[1] *Par sa signature, l'étudiant-e s'engage à respecter strictement la directive DI.1.2.02.07 liée au travail de diplôme.*
Durch seine Unterschrift verpflichtet sich der/die Student/in, sich an die Richtlinie DI.1.2.02.07 der Diplomarbeit zu halten.

Rapport reçu le / *Schlussbericht erhalten am* ..... Visa du secrétariat / *Visum des Sekretariats* .....



Bioluminescence

Diplômant Vincent Giachino

Objectif du projet

Dans le cadre du projet européen BRAAVOO, qui vise à mesurer les concentrations de différents polluants présents dans la mer, nous développons un lecteur de bioluminescence embarqué dans une bouée.

Méthodes | Expériences | Résultats

La méthode retenue utilise les propriétés de bioluminescence de micro-organismes. Différentes bactéries, génétiquement modifiées, sont séchées et placées dans des biochips comprenant dix puits indépendants les uns des autres.

Lorsqu'elles sont réactivées avec de l'eau, elles réagissent en émettant de la lumière dont la cinétique évolue au cours du temps, chacune différemment selon le type de polluant présent.

Le système de mesure utilise un capteur qui permet de quantifier la bioluminescence émise. Les photons captés par le détecteur sont comptés pendant dix secondes. Ce capteur se déplace en dessous de chaque puits du biochip, prend une mesure et recommence l'opération pendant 2 heures. Il transmet ensuite à l'opérateur, respectivement un ordinateur central, un tableau des valeurs acquises.

En étudiant l'évolution de la courbe caractéristique pour chaque puits, la concentration de polluant peut être déterminée.

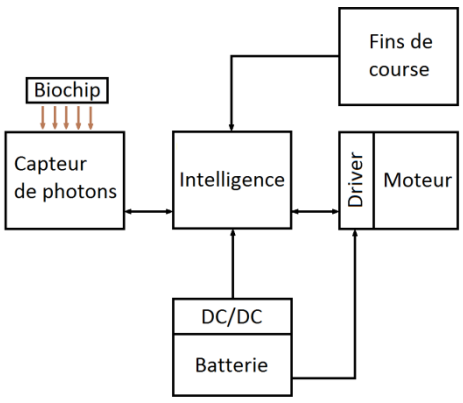
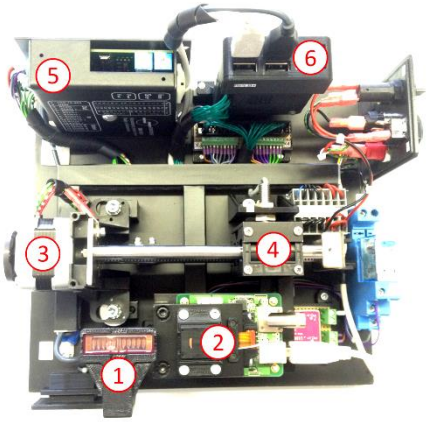


Schéma de principe du système de mesure.
Le micro-ordinateur, Raspberry Pi B+, gère les déplacements du moteur et commande le capteur de photons



1 : Support du biochip
2 : Capteur de photons
3 : Moteur pas à pas
4 : Chariot avec tige fileté
5 : Driver du moteur
6 : Raspberry Pi B+

Travail de diplôme
| édition 2015 |

Filière
Systèmes industriels

Domaine d'application
Power & Control

Professeur responsable
Prof. Martial Geiser
martial.geiser@hevs.ch

Partenaire
Department of Fundamental Microbiology, UNIL, 1015 Lausanne

Table des matières

1. Terminologie 2
2. Introduction 2
2.1 Généralités 2
2.2 Description de l'expérience 3
2.3 Schémas de principe 3
2.4 Système de mesure 6
3. Méthode & Matériel 7
3.1 Méthode 7
3.2 Matériel 8
3.3 Installation 10
3.4 Configuration du matériel 11
3.5 Principe d'utilisation du driver et du capteur uPMT 12
3.6 Conception mécanique 17
3.7 Conception électriques 18
3.8 Programmation 19
3.9 Mise en route 30
4. Résultats 32
4.1 Mesure de la consommation électrique moyenne 32
4.2 Mesure de la répétabilité de positionnement 33
4.3 Mesure dans l'obscurité (sans biochip) 34
5. Discussions 35
5.1 Mesure de la consommation électrique moyenne 35
5.2 Mesure de la répétabilité de positionnement 35
5.3 Mesure dans l'obscurité (sans biochip) 35
6. Conclusion & Améliorations 37
6.1 Conclusion 37
6.2 Améliorations & Travaux futurs 37
7. Références 40
7.1 Publications 40
7.2 Images 40
8. Annexes 40
9. Remerciements 42

Bioluminescence

1. Terminologie

DC : Direct courant
GND : Ground

USB : Universal serial bus
PCB : Printed circuit board

RPI : Raspberry Pi
GPIO : General programmable input/ouput
DI : Digital input
DO : Digital output

LSB : Least significant bit
MSB : Most significant bit

VID : Vendor identification
PID : Product identification

PAP : Pas à pas (moteur)
uPMT : Micro photomultiplier tubes

2. Introduction

2.1 Généralités

Ce travail de diplôme s'intègre dans le projet européen BRAAVOO qui vise à mesurer en temps réel et sur le site lui-même la concentration de différents types de polluants présents dans un milieu marin.

Des bactéries génétiquement modifiées, sélectionnées pour leurs réactions en présence de pollution, sont séchées et placées dans un petit contenant, nommé biochip.

Ce biochip possède dix puits indépendants les uns des autres, qui sont chacun remplis de bactéries différentes.

Une fois l'eau de mer ajoutée à chaque puits, selon le type de pollution présent dans l'eau, certaines bactéries vont réagir et émettre de la bioluminescence. Des polluants tels que les métaux lourds et des énergies fossiles peuvent être décelés facilement.

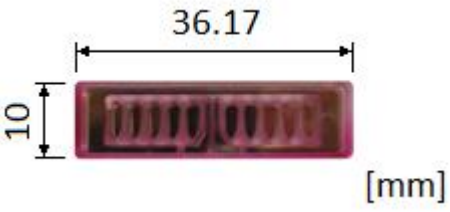


Figure 1 – Dimensions du biochip de mesure

La cinétique de la courbe d'émission des photons émise par les bactéries est de type exponentielle, durant les 2 à 3 premières heures. En définissant la constante de temps de l'équation de cette courbe, on peut en déduire la concentration de pollution présente.

2.2 Description de l'expérience

Une fois l'eau de mer ajoutée au biochip par un système de remplissage, des mesures sont prises sous chaque puits pendant dix secondes. Durant cette période, les photons émis sont comptés et la valeur de leur nombre est enregistrée.

Le système de mesure se déplace pour effectuer une mesure en dessous de tous les puits et recommence l'opération pendant 2 heures. A la fin le fichier créé est transmis à un ordinateur central qui collecte les données et effectue les calculs de concentration avec les résultats obtenus.

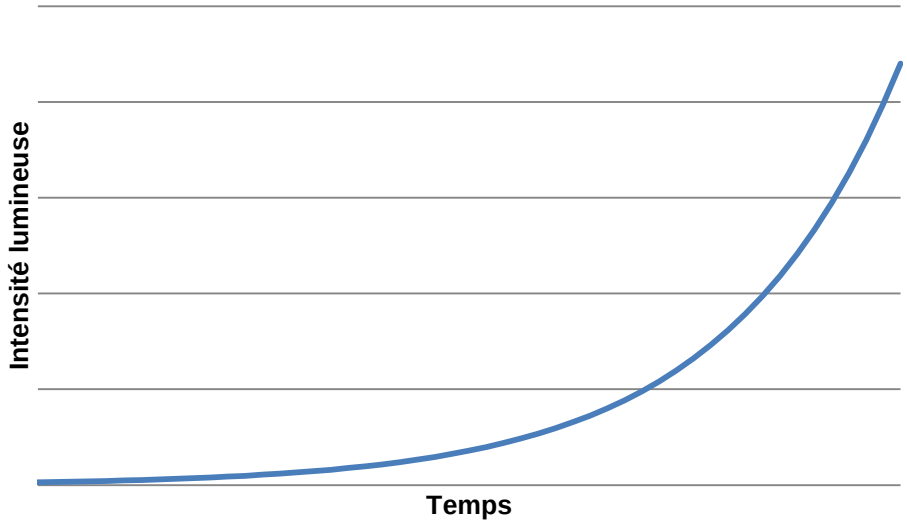


Figure 2 - Courbe typique d'émission de bioluminescence pendant les deux premières heures après l'ajout de polluant aux bactéries

2.3 Schémas de principe

Le système de mesure et de remplissage des biochips sont placés dans une bouée qui se situe dans la mer. Un magasin, qui contient une trentaine de biochips, permet au système d'être indépendant pendant un mois, à raison d'une mesure par jour.

Les mesures s'effectuent quotidiennement ou sur demande de l'opérateur qui contrôle l'ordinateur central.

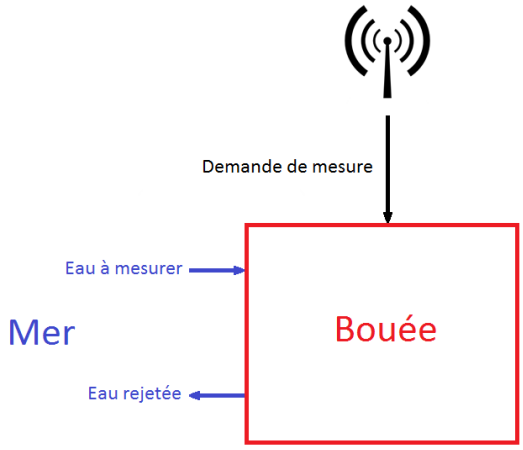


Figure 3 - Schéma de principe de la bouée

Des biochips sont placés dans une boite de réserve ou un magasin en forme de disque, pivotant sur lui-même *(encore à définir)*. A chaque mesure un système d'entrainement en déplace un à une position définie, pour qu'il puisse être rempli avec l'eau de mer.
Une fois l'opération de remplissage terminée, le système déplace le biochip à une deuxième position pour que les mesures puissent être effectuées.
Le capteur se déplace sous chaque puits du biochip, compte le nombre de photons captés pendant dix secondes, passe au puits suivant jusqu'au dernier et répète l'expérience pendant 2 heures. Il collecte les données et elles sont envoyées à l'ordinateur central.

Dans le cadre de ce projet, seul la partie du système de mesure sera fabriquée. Pour permettre d'utiliser cette partie de manière indépendante, les commandes, normalement transmises par l'ordinateur central, de démarrage de l'expérience et d'arrêt sont remplacées par un commutateur à deux positions « Start » et « Stop ».
Les données collectées pendant les deux heures sont, pour l'instant, stockées sur une clé USB, externe au système de mesure.

Ce projet comprend une boite étanche à la lumière dans laquelle se situera le système automatisé de mesure. Ce boitier comporte une ouverture en son sommet pour y placer le biochip. Un cache permet de la refermer et assurer une étanchéité à la lumière extérieure. Une commande avec un bouton, permettant de donner le démarrage de l'expérience et l'arrêt si nécessaire, et un port USB déporté qui permettra d'insérer une clé USB pour récolter les informations mesurées est également à disposition. Les données seront enregistrées dans un fichier, sous un format « .csv », qui peut être ouvert avec le logiciel Excel.

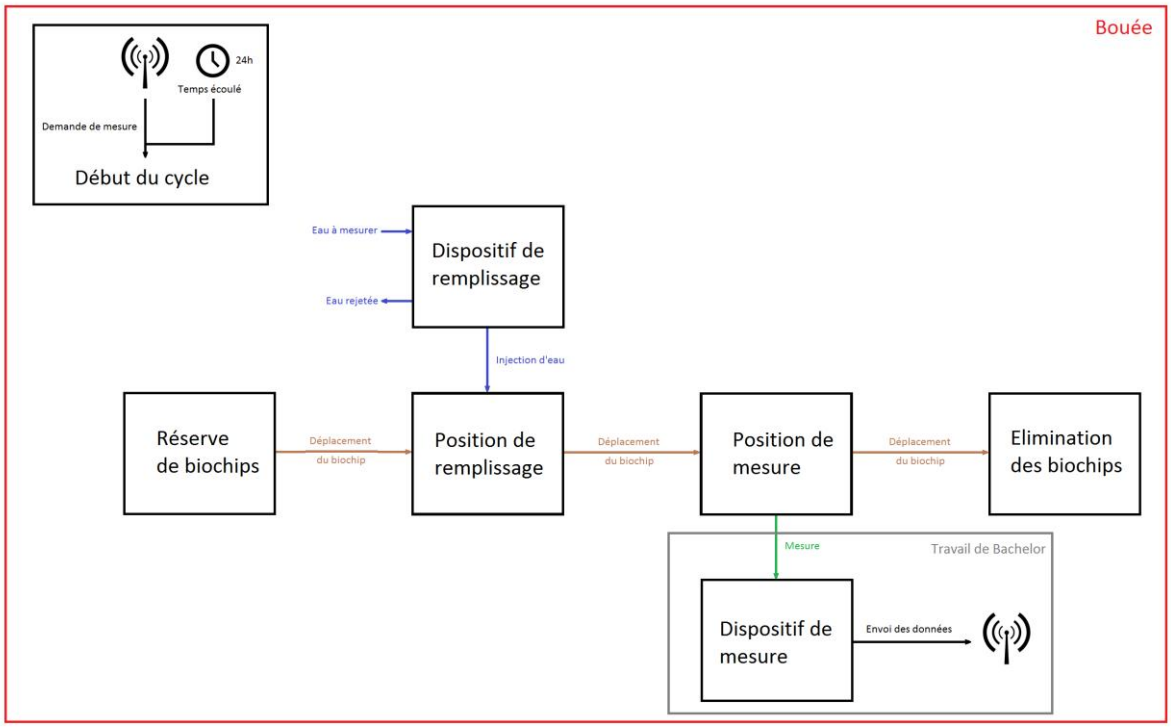


Figure 4 - Schéma de principe général à l'intérieur de la bouée

Les différentes parties du système peuvent se décomposer comme tel :

- **Début du cycle** : Il est donné par un ordinateur central situé sur les côtes qui informe le système du démarrage de l'expérience.

- **Réserve de biochips** : Un magasin, qui peut contenir une trentaine de biochips, permet au système d'être autonome pendant un mois. Les biochips sont déjà tous remplis de bactéries séchées qui seront réactivées une fois l'eau de mer ajoutée. Les puits du biochip contiennent différents types de bactéries, qui réagissent chacune aux différents polluants présents dans la mer.

- **Dispositif de remplissage** : Il permet au biochip d'être rempli d'eau de mer. Il comprend un tuyau d'amenée et un tuyau pour rejeter l'eau excédante qui n'est pas utilisée.

- **Position de remplissage** : C'est à cette position que le biochip va être rempli d'eau de mer.

- **Dispositif de mesure** : Il va effectuer la mesure pendant les deux heures. Le capteur de photons va se déplacer sous chaque puits du biochip en continu. Sous chacun d'entre eux, il compte le nombre de photons émis par les bactéries et ceci pendant dix secondes. Il enregistre ensuite toutes ces données sous un fichier au format « .csv » qu'il transmet ensuit à l'ordinateur central.

- **Position de mesure** : C'est à cette position que l'émission lumineuse va être mesurée pendant les deux heures que dure la mesure.

- **Elimination des biochips** : C'est un boite où les biochips qui ont déjà subit une mesure complète sont rejetés.

Le tout se situe dans la bouée. Une fois par mois, lorsque l'opérateur intervient, il vide la boite contenant les biochips usagés et rempli un nouveau magasin.

2.4 Système de mesure

Le système de mesure, réalisé dans ce projet, se compose d'une intelligence centrale qui gère tous les séquencements de l'expérience.

Un capteur de photons permet de mesurer les émissions de bioluminescence sous chaque puits du biochip. Ce capteur est livré avec son kit de démonstration qui permet de le commander. Il est relié via un connecteur USB au micro-ordinateur embarqué.

Pour effectuer les déplacements, un moteur PAP, avec une tige filetée reliée à son arbre et un chariot qui soutient le capteur, est utilisé. La commande de ce moteur se fait via un driver.

Deux fins de courses à galet permettent d'assurer que le moteur n'entre pas en collision avec les butées avant et arrière. Un détecteur de proximité inductif permet de prendre le « zéro » avant un le début d'une expérience.

Le biochip est placé sur un support à une position fixe. Elle peut cependant être réglée en hauteur et en profondeur, dans le sens perpendiculaire au déplacement du capteur de photons.

Le tout est alimenté par une batterie $24V_{dc}$. Pour ce projet, cette batterie est temporairement remplacée par une alimentation de laboratoire $24V_{dc}$. Pour alimenter le micro-ordinateur en $5V_{dc}$, un convertisseur DC/DC de 24V à 5V est employé.

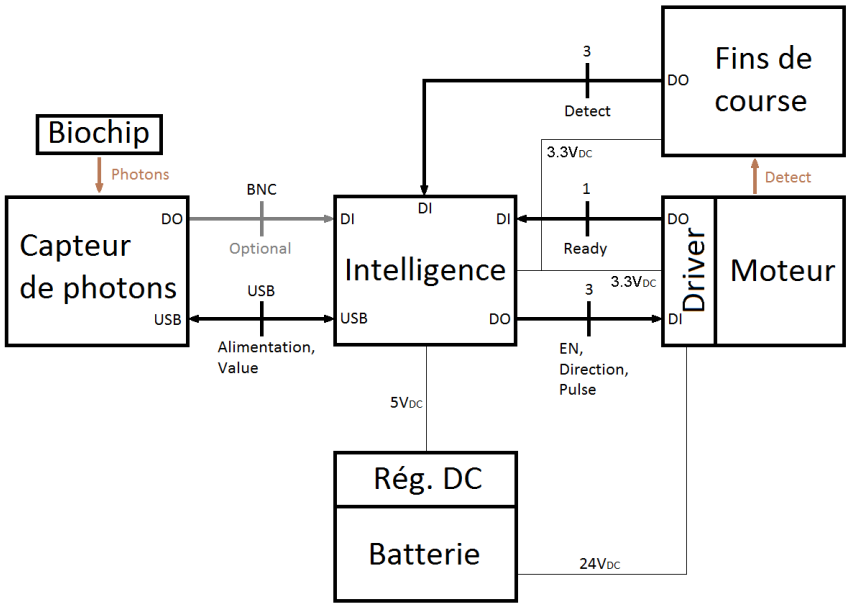


Figure 5 - Schéma de principe du système de mesure

3. Méthode & Matériel

3.1 Méthode

Le capteur de photons uPMT se déplace sous chacun des dix puits, lors d'une série de mesures. Ces déplacements s'effectuent en trois phases :

1) Dans une première phase, le système se déplace à la position « zéro », qui est donnée par le détecteur de proximité inductif.

2) Puis dans un second temps, il va rejoindre le premier puits du biochip, qui se trouve à 13.14mm de la position « zéro ». Cette valeur est donnée dans une variable du programme.
   Si le « zéro » est décalé, l'opérateur peut déplacer le détecteur de proximité inductif manuellement pour permettre au capteur uPMT de se retrouver idéalement placé sous les puits du biochip. Le support du détecteur de proximité inductif comprend un trou oblong, pour permettre le déplacement longitudinal, dans le sens du mouvement du chariot.

3) Ensuite le système va se déplacer sous tous les puits du biochip. Ils sont séparés de 2.54mm, sauf entre le cinquième et le sixième où il y a 5.08mm entre eux.
   Le système revient chaque fois à la première position, à la fin d'un cycle, et recommence une série de mesures, et ceci pendant 2 heures.

Lors d'une série de mesures, l'opérateur peut également prendre le contrôle du système et effectuer une phase de calibration pour prendre le « zéro », à la place du *point 1*. Dans ce cas, le micro-ordinateur prendra cette référence pour situer le premier puits du biochip et n'effectuera pas le *point 2*.

La précision lors des déplacements doit permettre à la tête de mesure du capteur uPMT de se situer toujours sous la surface du biochip. La tête de mesure est un rectangle de 1x3mm. Le puits vaut dans sa plus petite largeur 1.4mm.
La tolérance correspond donc à la formule suivante :

$$L_{tolérance} = \frac{L_{puits,min} - L_{têtedemesure}}{2} = \frac{1.4 - 1}{2} = 0.2\text{mm}$$

En ce qui concerne le positionnement, la tolérance acceptée est de ±0.2mm.
Elle doit comprendre la répétabilité de positionnement au « zéro », qui est fixée à ±0.1mm, c'est-à-dire que le détecteur de proximité inductif doit déclencher dans une plage de ±0.1mm.
Elle doit tenir également compte de la tolérance du déplacement en lui-même qui est fixée aussi à ±0.1mm. Elle englobe la distance minimale par pas, respectivement par divisions de pas, et le jeu de la tige filetée avec le chariot mobile.

3.2 Matériel

Le matériel utilisé dans le cadre de ce projet se compose de :

- **Intelligence** (micro-ordinateur) : Raspberry Pi (RPI) B+
- **Driver** : SMD 9103 SONCEBOZ
- **Moteur PAP** (avec système d'entrainement linéaire) : 8453 SONCEBOZ
- **Capteur de photons** : uPMT HAMAMATSU
- **Kit de démonstration** : Evaluation kit uPMT EK1 HAMAMATSU
- **Fin de course à galet** : SS-5GL2
- **Détecteur de proximité inductif** : IFRM 08P1701 L BAUMER
- **Relais 24V_{dc}** : 40.61 FINDER

Les datasheets des composants utilisés ont été mis en annexe. ***[Annexes : 5.1 à 5.8]***

Depuis le RPI, la commande du capteur de photons uPMT se fait via le port USB.
Pour commander le driver du moteur, il faut trois sorties. La première pour activer la commande du moteur, la deuxième pour choisir le sens de rotation et la dernière pour envoyer une impulsion qui fait avancer le moteur PAP d'un pas, respectivement d'une division de pas.
Le RPI comporte également deux sorties qui allume deux leds pour indiquer à l'opérateur le mode de fonctionnement et la demande d'un « Start ».

Il y a aussi six entrées utilisées sur le RPI, deux d'entre elles pour démarrer et réinitialiser le programme à l'aide d'un bouton à trois positions.
Il y a également trois entrées pour disposer des informations sur la position du chariot, deux sont utilisées pour les fins de courses et une pour le détecteur de proximité inductif pour la mise à « zéro ».
Le driver dispose également d'une sortie qui vient connectée sur une entrée du RPI qui s'active lorsque le moteur est « prêt » pour un déplacement.

L'alimentation en 24V_{dc} est câblée au driver, le détecteur de proximité inductif et sur le primaire du convertisseur DC/DC afin d'abaisser le potentiel à 5V_{dc} pour alimenter le RPI.
Le RPI fourni 3.3V_{dc} qui va alimenter sur les fins de courses, le contact du relais qui est activé par le détecteur de proximité inductif et le contact du driver qui informe l'opérateur sur l'état du moteur.

Le détecteur de proximité inductif sert à prendre le « zéro » avant de commencer une série de mesures. Ce capteur doit offrir une répétabilité de l'ordre de ±0.1mm.

Dans le catalogue de BAUMER, l'information de répétabilité est indiquée comme tel :

> … *« La reproductibilité du point de commutation effectif, établie sur deux mesures quelconques dans l'espace de 8 heures, est de ±5%. Ceci, pour une plage de température comprise entre +15°C et +30°C et pour une tension d'alimentation dont les variations sont inférieures à ±5%. »* … ***[Réf. 1.1]***

Le détecteur utilisé est le IFRM 08P1701 L. Sa portée effective est de 2mm. Ce qui donne une valeur de répétabilité calculée grâce à la formule suivante :

$$L_{répétabilité} = S_n \times S_{tolérance} = 2 \times 0.05 = 0.1mm$$

, avec S_n : Distance effective [mm]
$S_{tolérance}$: Tolérance de répétabilité [%]

Cette valeur correspond à la valeur de tolérance imposée pour que la tête de mesure du capteur de photons uPMT se situe correctement sous le puits du biochip.

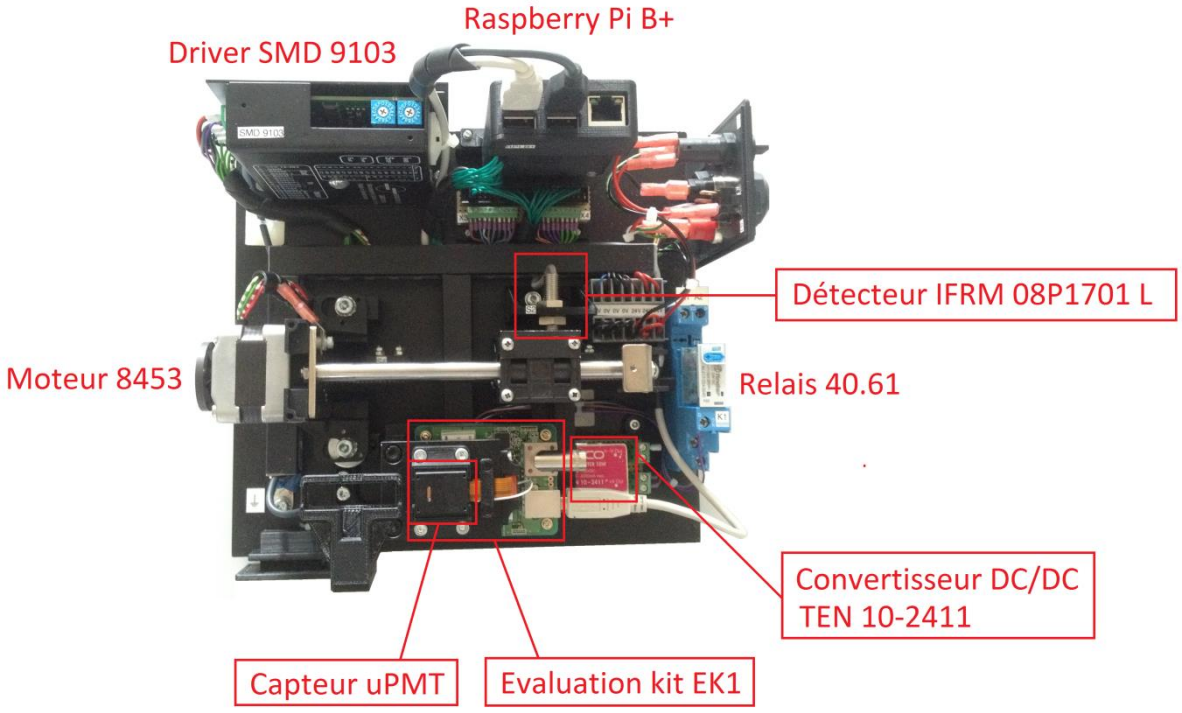


Figure 6 – Vue de dessus du système complet (sans le capot)

3.3 Installation

Comme les mesures se font dans l'obscurité la plus absolue, les leds sur le RPI et le driver SMD 9103 doivent être dessoudées.

Il y a deux leds sur le RPI qui indique son fonctionnement et la présence de la carte micro SD.

Il y a également deux leds sur le driver SMD9103 pour indiquer son état de marche et l'éventualité de défauts présents.

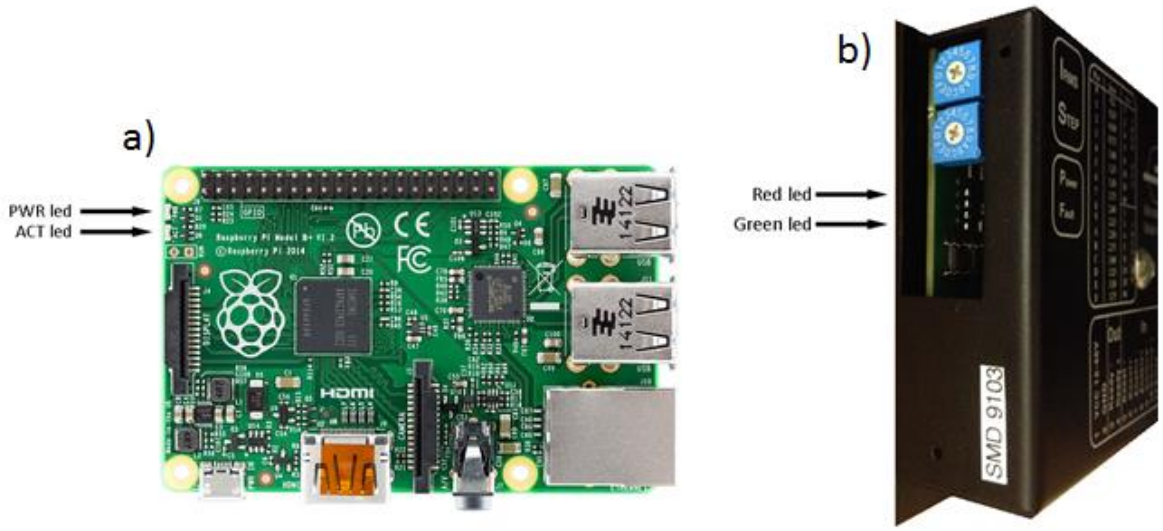


Figure 7 – a) Emplacement des leds sur le RPI *[Réf. 2.1]*
b) Emplacement des leds sur le driver SMD 9103

3.4 Configuration du matériel

3.4.1 Configuration du driver SMD 9103

Deux paramètres peuvent être réglés sur le driver à l'aide de deux commutateurs.

Le premier permet de choisir le courant par phase du moteur.
Le moteur utilisé consomme 0.34A/phase. Il est possible de fonctionner avec un courant réduit si la charge à déplacer n'est pas importante, comme c'est le cas pour notre projet où la charge fait moins de 150 grammes.
Le commutateur du courant « I_{RMS} » est donc placé sur « 0 » qui correspond à 0.2A/phase, comme stipulé dans la tabelle du driver SMD 9103.

Le deuxième permet de choisir le nombre de division par pas. Donc combien de pulses il faut envoyer au driver pour que le moteur effectue un pas. En effet, en jouant sur les niveaux de tensions appliquées aux bobines du moteur, il peut se positionner à différents endroits entre deux pas.
Une division de 10 entre deux pas offre un ratio entre précision et rapidité convenable.
La position du commutateur des divisions « Step » à été placée sur « D » qui correspond à 10 divisions, comme stipulé dans la tabelle du driver SMD 9103.

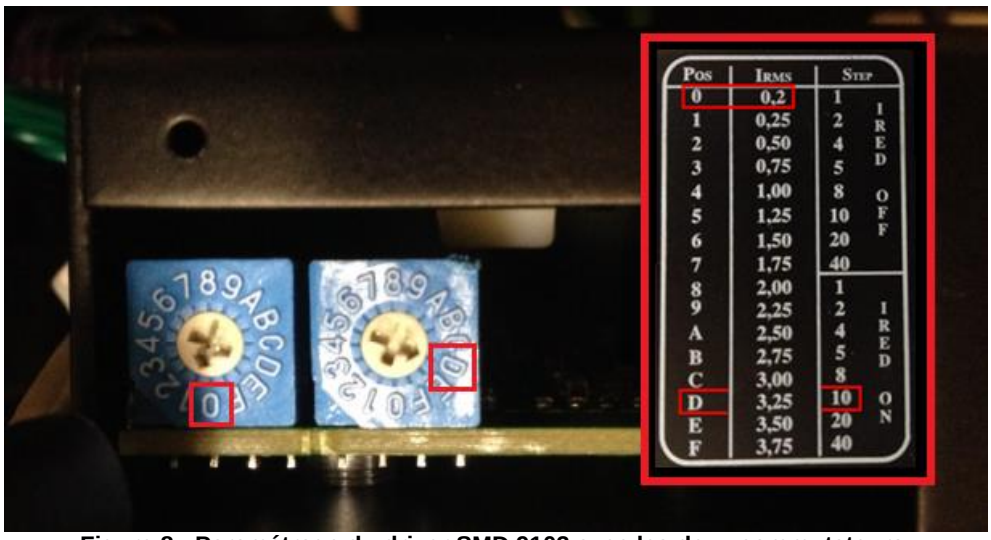


Figure 8 - Paramétrage du driver SMD 9103 avec les deux commutateurs.
Le commutateur « Step » sur « D »
Le commutateur « I_{RMS} » sur « 0 »

La limite théorique, selon le datasheet du fabricant, de fréquence de commutation pour les pulsations est de 200kHZ, mais à cette fréquence le moteur saute régulièrement des pas avec une charge. La fréquence choisie après plusieurs tests en conditions réelles, dans un laboratoire, est de 2kHz.

La vitesse de déplacement est donnée par la formule suivante :

$$v_{déplacement} = \frac{\frac{f_{pulse}}{2} \times Pas_{mécanique}}{N_{pulse} \times N_{division}} = \frac{\frac{2000}{2} * 0.002}{200 * 10} = 0.001 \frac{m}{s} = 1 \frac{mm}{s}$$

, avec f_{pulse} : Fréquence des pulsations envoyées au driver [Hz]
$Pas_{mécanique}$: Longueur du pas de la tige filetée [m]
N_{pulse} : Nombre de pulses par tour du moteur [-]
$N_{division}$: Nombre de divisions pour avancer d'un pas (driver) [-]

3.5 Principe d'utilisation du driver et du capteur uPMT

3.5.1 Principe d'utilisation du driver SMD 9103

L'alimentation du driver doit être comprise entre $18V_{dc}$ et $30V_{dc}$. Avec un point de fonctionnement optimal à $24V_{dc}$. Il est donc alimenté avec une tension de $24V_{dc}$, directement fournie depuis les batteries embarquées.

La commande du driver SMD 9103 s'effectue à l'aide de 3 entrées.
La première sert à enclencher la commande du moteur, la deuxième à gérer le sens de rotation du moteur et la dernière à avancer d'un pas à chaque fois qu'une impulsion lui est envoyée.
Si une division du nombre de pas a été paramétrée avec le commutateur du driver, une impulsion sur l'entrée fait avancer le moteur d'une division de ce pas.
La fréquence des pulsations a été fixée à 2kHz, après différents essais, afin d'éviter de « sauter » un pas lors d'un déplacement et par conséquent de perdre la position.

Le driver commande le moteur PAP avec 4 sorties. Elles correspondent aux branchements des deux bobines de celui-ci. Il s'agit d'un moteur bipolaire, les bobines n'ont pas de point milieu et sont donc alimentées avec deux fils.

Si le moteur relié au driver est prêt et peut tourner, une sortie permet de renvoyer cette information. Un contact se ferme entre la borne 4, qui est alimentée en 3.3V, et la borne 3, qui est reliée au RPI.

Pin N°	Signal	Description
1	VCC +	Supply 12 (18) – 40 Vdc
2	GND	
3	Ready -	No default output
4	Ready +	
5	Boost - (0 V)	Motor current boost input
6	Boost + (5 V)	
7	Enable - (0 V)	Motor current on input
8	Enable + (5 V)	
9	CW/CCW - (0 V)	Direction of rotation input
10	CW/CCW + (5 V)	
11	Pulse - (0 V)	Step pulse input
12	Pulse + (5V)	
13	Phase B2	Phase B of motor
14	Phase B1	
15	Phase A2	Phase A of motor
16	Phase A1	

Tableau 1- Plan du bornier du driver SMD 9103 *[Réf. 2.2]*

3.5.2 Principe d'utilisation du capteur de photons uPMT

Le capteur uPMT de HAMAMATSU compte le nombre de photons qui traverse sa tête de mesure. Elle fait 1x3mm. Pour pouvoir effectuer les mesures, le capteur à besoin d'une tension de 1'000V délivrée par un module qui est fourni avec le capteur.
Ce capteur peut percevoir des longueurs d'onde allant de 280nm à 600nm.

Le capteur est livré avec sont kit de démo (evaluation kit uPMT EK1). Il s'agit d'un système électronique monté sur un PCB avec un microprocesseur pour le contrôle et une FPGA pour les calculs.
Ce kit permet d'enclencher, d'éteindre et de paramétrer le temps d'intégration le capteur uPMT, c'est-à-dire la période pendant laquelle le comptage du nombre de photons va être effectuée.

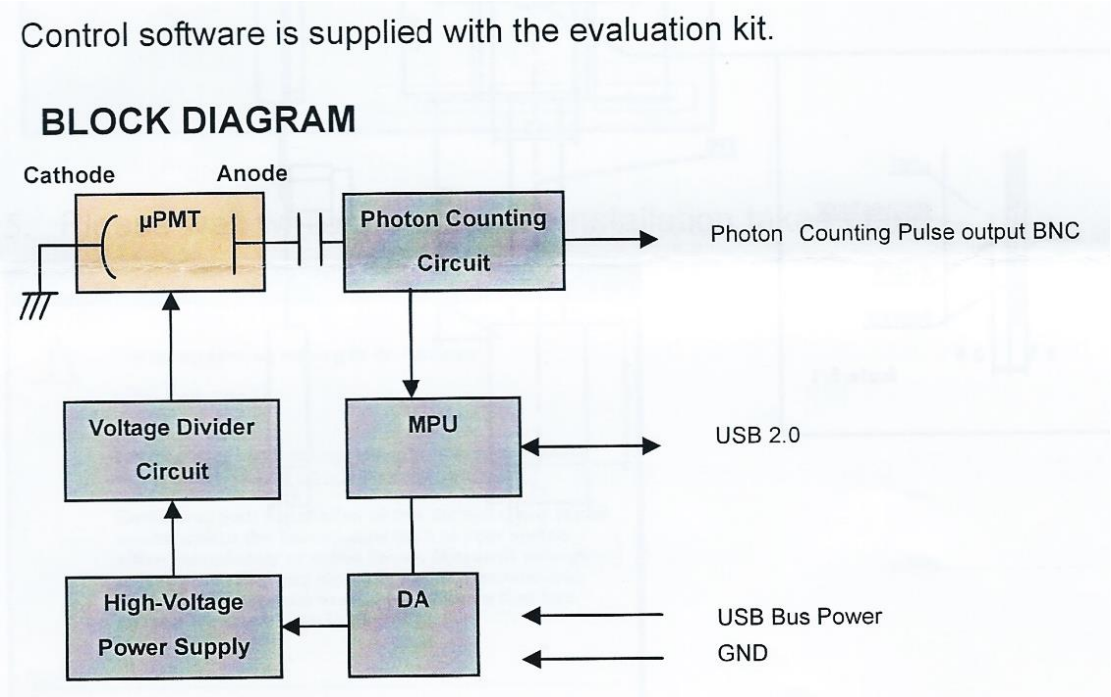


Figure 9 - Diagramme de contrôle du capteur uPMT avec le kit et le software de HAMAMATSU *[Réf. 2.3]*

Le contrôle du capteur s'effectue avec un programme (software), fourni par le fabricant HAMAMATSU, qui s'exécute sous Windows 32-bits et qui communique avec le kit de démonstration avec un câble USB.
Comme le RPI fonctionne sous Linux, respectivement Raspbian, le programme développé par HAMAMATSU ne peut pas être utilisé.

Pour palier à ce problème, les trames échangées entre le programme et le capteur sont « snoopées ». C'est-à-dire qu'elles sont visualisées par un software. En l'occurrence, « SnoopyPro » qui est un « sniffer USB ». Ce programme collecte les paquets échangés depuis l'ordinateur jusqu'au capteur et vice-versa.

Une trame envoyée depuis l'ordinateur peut valoir de 1 à 8 bytes. Les 4 premiers définissent la fonction choisie et les 4 derniers la valeur, si celle-ci en nécessite une. Cette valeur commence par le LSB et finie par le MSB.
Chaque fois qu'une commande est envoyée au capteur, celui-ci renvoie un paquet de 64 bytes en réponse.

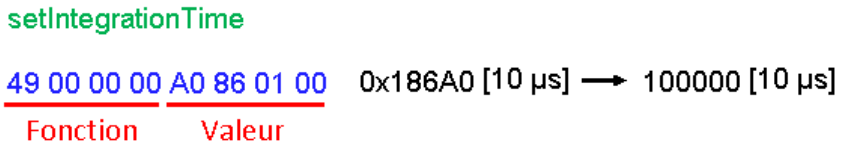


Figure 10 - Exemple de commande envoyée au capteur uPMT « setIntegrationTime » avec une valeur de 10s (0x186A0 µs)

Les commandes pour l'activation et la désactivation du capteur sont ordrées comme suit (avec les paquets envoyés) :

Activation :

- **setDefinedHV** : Active la haute tension (1'000V) [V]
  - `(0x 44 56)`
- **setDefinedLLD** : Active la basse tension (376mV) [mV]
  - `(0x 44 4C)`
- **setIntegrationTime** : Défini le temps d'intégration (10s dans ce projet) [µs]
  - `(0x 49 00 00 00 A0 86 01 00)`
- **setRepeat** : Permet au capteur de compter en continu
  - `(0x 52)`
- **StartCount** : Démarre le comptage
  - `(0x 43)`

Désactivation :

- **StopCount** : Arrête le comptage
  - `(0x 0D)`
- **setHV** (avec une valeur de 0V) : Désactive la haute tension (0V) [V]
  - `(0x 56 00 00 00 00 00 00 00)`

Après l'activation, le capteur envoie la valeur du nombre de photons mesurés pendant dix secondes. On reçoit un paquet de 64 bytes en retour en hexadécimal avec le LSB en premier et le MSB en dernier.

Lecture de tout les bytes sauf les 4 premiers (fonction)

```
for (int i = 0; i < (dataLength - headerLength); i++){
    if (dataTemp[i] > 0) {
        tempValue += (double) (dataTemp[i] * pow(256.0, i))
    }
}
return tempValue;
```

Convertion des valeurs hexadécimales en un entier

Figure 11 - Conversion des paquets reçus en nombre entier

Les 4 premiers bytes donnent la fonction et les 60 suivants indiquent la valeur en hexadécimale. Pour transformer cette valeur en un nombre réel, il faut utiliser la formule suivante :

$$\mathrm{Value}_{\mathrm{int}} = \mathrm{Value}_{\mathrm{hex}} \times 256^{\mathrm{Position}}$$

, avec $\mathrm{Value}_{\mathrm{hex}}$: Valeur du nombre en hexadécimal
Position : Position de ce nombre dans la trame (de 0 à 59)

Il faut ensuite effectuer la somme de toutes les valeurs pour obtenir le nombre exact de photons qui ont parcourus le capteur pendant les dix secondes qu'a durée la mesure.

Des paramètres importants, pour l'initialisation du périphérique USB, sont les identifications du vendeur et du produit, qui sont propre à chaque matériel. Il est aussi nécessaire de connaître le « endpoint » pour les transferts et les réceptions des données.
Grâce au « snooping » de l'échange entre l'ordinateur et le capteur ces valeurs ont pu être trouvées.

Voici les valeurs pour le capteur de photons uPMT avec son evaluation kit (uPMT EK1) :

- **VID** : 0x0661
- **PID** : 0x3706
- **Endpoint** : 0x01

Un tableau Excel contenant les informations sur les paquets envoyés et reçus est mis en annexe. ***[Annexe : 6.1]***

3.6 Conception mécanique

L'ensemble des composants utilisés sont fixés sur un châssis. Comme le projet doit prendre le moins de place possible, les dimensions de celui-ci sont 220x240mm avec une épaisseur de 10mm.

Pour assurer l'étanchéité lumineuse du système, un couvercle d'une hauteur d'environ 150 mm peut être emboité et visé au châssis. Une ouverture permet d'insérer le biochip sur un support qui est fixé sur le châssis.
Ce support peut être réglé en hauteur. Il peut également être déplacé dans le sens perpendiculaire au mouvement du capteur de photons.

Les trois pieds du moteur PAP, avec son chariot, étant situés à différentes hauteur, reposent sur des plots qui sont vissés sur le châssis. Sur le chariot qu'entraine la tige filetée, solidaire de l'arbre du moteur, est placé le capteur de photons uPMT. Il repose sur un support conçu avec des plaques d'aluminium et des pièces imprimées en 3D.

Voir ***[Annexe : 3.1.2]***

Les fins de courses à galet sont également fixés sur des supports qui sont placé sur le châssis. Deux trous oblongs fraisés dans les supports permettent de régler la hauteur de ces fins de courses. Grâce à eux, on peut ainsi réduire la course maximale entre les positions maximales IN et OUT.

Le support du détecteur de proximité inductif est imprimé en 3D. A sa base, un trou oblong est fait pour permettre l'ajustage longitudinal du détecteur. Le relais que fait tirer le détecteur est placé sur un rail DIN.

Le driver SMD 9103 et le RPI sont directement montés sur le châssis. Pour le RPI, un boitier a été imprimé en 3D.

Les borniers ainsi que le convertisseur DC/DC sont « clipsés » dans des supports qui sont fixés à l'aide de vis au châssis.

Les schémas techniques des pièces usinées sont donnés en annexe. ***[Annexes : 3.2.1 à 3.5.2]***
Un dessin des pièces imprimées en 3D est également mis en annexe. ***[Annexe : 4.1]***

3.7 Conception électriques

Les schémas électriques renseignent sur le branchement de tous les composants utilisés. Deux condensateurs de 220µF/50V polarisés ont été ajoutés avant et après le convertisseur DC/DC. Le premier sur le $24V_{dc}$ et le deuxième sur le $5V_{dc}$.
Ils permettent d'éviter de perdre l'alimentation du RPI, si pour une raison le moteur venait à consommer d'avantage pendant un cours laps de temps.

Voir ***[Annexes : 1.1 et 1.2]***

Comme le courant est en dessous de 1A, en fonctionnement normal, la section des fils a été fixée à $0.25mm^2$.

Différentes couleurs de fils ont été utilisées, elles se séparent comme tel :

- ● Rouge : $24V_{dc}$
- ● Brun : $5V_{dc}$
- ● Gris : $3.3V_{dc}$
- ● Noir : GND
- ●● Jaune/vert : Terre

- ●○ Vert + Vert/blanc : Bobine A du moteur PAP
- ●○ Rouge + Rouge/blanc : Bobine B du moteur PAP

- ● Violet : Entrée RPI ($3.3V_{dc}$)
- ● Vert : Sortie RPI ($3.3V_{dc}$)

- ● Jaune : Commande « Start » (déportée)
- ● Rose : Commande « Stop » (déportée)
- ● Bleu : GND led « State » (déportée)
- ○ Blanc : GND led « Start » (déportée)

3.8 Programmation

3.8.1 Code

La programmation a été faite en C++.

Voir ***[Annexes : 2.1 à 2.5]***

Parmi les librairies de base du C++, les suivantes ont été utilisées :

- **<iostream>** : Librairie pour gérer les entrées et sorties de la console
- **<math.h>** : Librairie pour exécuter des fonctions mathématiques
- **<string.h>** : Librairie pour créer des chaînes de caractères
- **<time.h>** : Librairie pour utiliser la date et le temps
- **<sstream>** : Librairie utilisée pour effectuer des modifications sur des « string »
- **<fstream>** : Librairie utilisée pour effectuer des modifications sur des « string »

En plus de ces libraires, deux ont dues être installées dans le RPI pour pouvoir utiliser les entrées et sorties et pour utiliser le port USB. Il s'agit de :

- **<wiringPi.h>** : Librairie qui permet de paramétrer et d'utiliser les entrées et sorties du RPI
- **<libusb.h>** : Librairie qui permet d'utiliser le port USB, d'envoyer et recevoir des paquets

Dans la programmation liée à notre application, trois classes ont été créées. Celles-ci sont :

- **<Bioluminescence.cpp>** : Cette classe contient le « main » du programme. Elle sert à effectuer les appels des autres classes. C'est elle qui gère le déroulement de l'expérience et les interfaçages avec l'opérateur.
- **<Motor.cpp>** : Cette classe gère les déplacements du moteur. C'est également elle qui lit toutes les entrées et écrit sur toutes les sorties du RPI.
- **<USBTransfer.cpp>** : Cette classe gère la communication par le port USB du RPI. Elle génère la liaison avec un périphérique, crée les messages à envoyer et lit les paquets reçus.

Dans chacune de ces classes, des méthodes servent à réaliser une tâche précise.
Hormis pour la classe <Bioluminescence.cpp> qui ne contient que le « main ». Elle n'a pas d'autres méthodes mais elle s'occupe d'appeler les méthodes des autres classes.

Les différentes méthodes utilisées dans le code sont :

- **<Motor.cpp>** :
    - **startEnable()** : Renvoi la valeur du bouton « Start »
        - Retour 0 : Pas activé
        - Retour 1 : Activé

- **stopEnable()** : Renvoi la valeur du bouton « Stop»
  - Retour 0 : Pas activé
  - Retour 1 : Activé

- **turnOnStateLed(period, blinking)** : Allume ou fait clignoter la led verte
  - period : *(int)* Donne la période de clignotement en millisecondes
  - Blinking : *(bool)* Gère le clignotement
    - Mise à 0 : Pas de clignotement
    - Mise à 1 : Clignotement

- **turnOnStartLed()** : Allume la led orange

- **turnOffStateLed()** : Eteint la led verte

- **turnOffStartLed()** : Eteint la led orange

- **movePositionZero()** : Déplace le capteur uPMT en position « zéro », donnée par le détecteur de proximité inductif
  - Retour 0 : Erreur lors du déplacement
    - Le moteur n'est pas prêt (entrée 40 du RPI à 0)
    - Un « reset » est demandé pendant la séquence (entrée 38 du RPI à 1)
  - Retour 1 : Positionnement correcte

- **move(distance, side, calibration)** : Déplace le capteur uPMT de la distance désirée dans le sens demandé
  - distance : *(int)* Donne la distance du déplacement en millimètres
  - side : *(bool)* Gère le sens du déplacement
    - Mise à 0 : Avance
    - Mise à 1 : Recule
  - Calibration : *(bool)* Indique le mode lors du déplacement
    - Mise à 0 : Pas en mode de calibration
    - Mise à 1 : En mode de calibration
  - Retour 0 : Erreur lors du déplacement
    - Le capteur de fin de course IN s'est activé (entrée 29 du RPI à 0)
    - Le capteur de fin de course OUT s'est activé (entrée 31 du RPI à 0)
    - Le moteur n'est pas prêt (entrée 40 du RPI à 0)
    - Un « reset » est demandé pendant la séquence (entrée 38 du RPI à 1)
  - Retour à 1 : Positionnement correcte

- **move(pulse, side)** : Déplace le capteur uPMT de la distance désirée dans le sens demandé
  - pulse : *(int)* Donne le nombre de pulses qui seront envoyées au driver
  - side : *(bool)* Gère le sens du déplacement
    - Mise à 0 : Avance
    - Mise à 1 : Recule
  - Retour 0 : Erreur lors du déplacement
    - Le capteur de fin de course OUT s'est activé (entrée 29 du RPI à 0)
    - Le capteur de fin de course OUT s'est activé (entrée 31 du RPI à 0)

- Le moteur n'est pas prêt (entrée 40 du RPI à 0)
- Un « reset » est demandé pendant la séquence (entrée 38 du RPI à 1)

▪ Retour à 1 : Positionnement correcte

- **<USBTransfer.cpp>** :

  - **initUSB()** : Effectue la séquence pour initialiser le port USB avec le capteur uPMT
    - Retour 0 : Erreur lors de l'initialisation
      - La clé USB n'est pas branchée au RPI
      - La clé USB n'a pas été branchée au démarrage du RPI
      - La clé USB n'est pas du bon format (VFAT)
    - Retour 1 : Initialisation réussie

  - **sendData(fonction)** : Envoi les paquets sur le port USB qui correspondent à la fonction choisie
    - fonction : *(int)* Permet de choisir quelle commande envoyer au capteur uPMT
      - Mise à 0 : « setDefinedHV » Active la haute tension (1'000V) [V]
      - Mise à 1 : « setDefinedLLD » Active la basse tension (376mV) [mV]
      - Mise à 2 : « setIntegrationTime » Défini le temps d'intégration (10s dans ce projet) [µs]
      - Mise à 3 : « setRepeat » Permet au capteur de compter en continu
      - Mise à 4 : « StartCount » Démarre le comptage
      - Mise à 5 : « StopCount » Arrête le comptage
      - Mise à 6 : « setHV » (avec une valeur de 0V) Désactive la haute tension (0V) [V]

  - **readData(data[ ], dataLength)** : Lit les paquets envoyés depuis le périphérique
    - data[] : *(unisgned char)* Permet de récolter les bytes dans un tableau
    - dataLength : *(int)* Indique la taille du tableau data[]

  - **sequenceOfTransfer(sensor, headerLength, dataLength)** : Crée une séquence de mesure complète, en lecture et en écriture, pour le capteur de photons uPMT
    - sensor : *(USBTransfer)* Donne le lien vers le port USB
    - headerLength : *(int)* Indique la taille de l'entête de la trame à recevoir
    - dataLength : *(int)* Indique la taille du tableau data[]
    - Retour value : *(int)* Retourne la valeur de nombre de photons comptés pendant les dix secondes

  - **releasedUSB()** : Relâche la connexion USB
    - Retour 0 : Erreur lors du relâchement de la clé USB
    - Retour 1 : Relâchement de la clé USB réussi

3.8.2 Paramétrage des entrées/sorties du RPI

Les 40 pins du RPI sont séparées comme tel :

- 2 x 5V
- 2 x 3.3V
- 8 X GND
- 17 x GPIO
- 2 x I2C interface
- 5 x SPI interface
- 2 x UART interface
- 2 x ID EEPROM interface

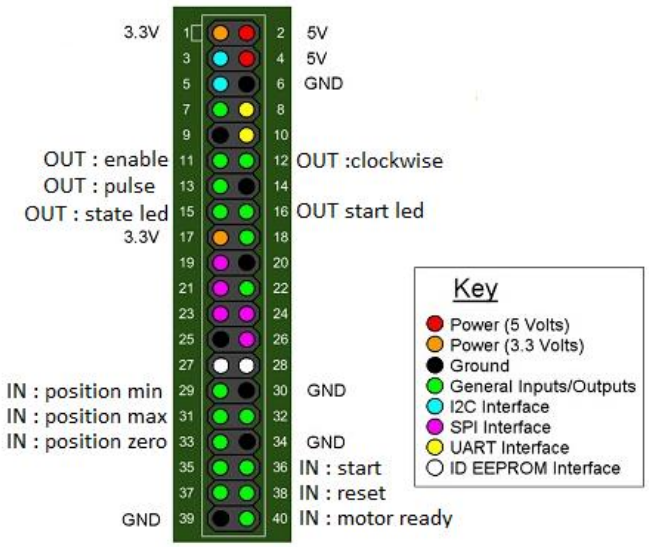


Figure 12 – Disposition des pins du RPI *[Réf. 2.5]*

Les GPIO peuvent être configurées pour être des entrées (0 à 3.3V) ou des sorties (0 à 3.3V).
Pour les paramétrées, il faut utiliser la librairie <wiringPi >.
Avec la méthode « pinMode », il est possible de définir si le pin concerné est une entrée (INPUT) ou une sortie (OUTPUT). Pour relier une entrée à une pull-up, une pull-down ou, par défaut, à une haute impédance, il faut utiliser la méthode « pullUpDnControl ».
Avec la méthode « digitalWrite » on peut donner une valeur à la sortie (LOW ou HIGH).

Dans le projet les sorties sont initialisées à 0V et le entrées sont connectées à des pull-down.
Il faut six entrées, deux pour la commande (« Start » et « Reset »), trois pour les positionnements (« max IN », « max OUT » et « position zero ») et une dernière pour que le driver renvoi l'information lorsque le moteur est prêt.
Il faut également cinq sorties, trois pour le driver du moteur (« enable », « clockwise » et « pulse ») et deux pour les affichages avec les leds (« State » et « Start »).

```
24      /*Pin number declaration*/
25      enableMove = 0;
26      clockwiseMove = 1;
27      pulseForMove = 2;
28      ledState = 3;
29      ledStart = 4;
30
31      startProgram = 27;
32      resetProgram = 28;
33      motorReady = 29;
34      positionMaxIn = 21;
35      positionMaxOut = 22;
36      positionZero = 23;
37
38      /*RPI configurations*/
39      wiringPiSetup();
40
41      pinMode(enableMove, OUTPUT); //Pin 11 RPI -> Output
42      pinMode(clockwiseMove, OUTPUT); //Pin 12 RPI -> Output
43      pinMode(pulseForMove, OUTPUT); //Pin 13 RPI -> Output
44      pinMode(ledState, OUTPUT); //Pin 15 RPI -> Output
45      pinMode(ledStart, OUTPUT); //Pin 16 RPI -> Output
46
47      pinMode(startProgram, INPUT); //Pin 36 RPI -> Input
48      pinMode(resetProgram, INPUT); //Pin 38 RPI -> Input
49      pinMode(motorReady, INPUT); //Pin 40 RPI -> Input
50      pinMode(positionMaxIn, INPUT); //Pin 29 RPI -> Input
51      pinMode(positionMaxOut, INPUT); //Pin 31 RPI -> Input
52      pinMode(positionZero, INPUT); //Pin 33 RPI -> Input
53
54      digitalWrite(enableMove, LOW); //Set the output value "LOW" to the pin 11
55      digitalWrite(clockwiseMove, LOW); //Set the output value "LOW" to the pin 12
56      digitalWrite(pulseForMove, LOW); //Set the output value "LOW" to the pin 13
57      digitalWrite(ledState, LOW); //Set the output value "LOW" to the pin 15
58      digitalWrite(ledStart, LOW); //Set the output value "LOW" to the pin 16
59
60      pullUpDnControl(startProgram, PUD_DOWN); //Set a pull-down to the pin 36
61      pullUpDnControl(resetProgram, PUD_DOWN); //Set a pull-down to the pin 38
62      pullUpDnControl(motorReady, PUD_DOWN); //Set a pull-down to the pin 40
63      pullUpDnControl(positionMaxIn, PUD_DOWN); //Set a pull-down to the pin 29
64      pullUpDnControl(positionMaxOut, PUD_DOWN); //Set a pull-down to the pin 31
65      pullUpDnControl(positionZero, PUD_DOWN); //Set a pull-down to the pin 33
66
```

Figure 13 – Code servant à paramétrer les entrées/sorties du RPI avec la librairie <wiringPi>

L'emplacement des pins est donné par le tableau de déclaration de la librairie <wiringPi> grâce à la commande « `gpio readall` » dans la commande du RPI.

```
pi@raspberrypi ~/WiringPi $ gpio readall
 +-----+-----+---------+------+---+---Pi 2---+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
 |   2 |   8 |   SDA.1 |   IN | 1 |  3 || 4  |   |      | 5V      |     |     |
 |   3 |   9 |   SCL.1 |   IN | 1 |  5 || 6  |   |      | 0v      |     |     |
 |   4 |   7 | GPIO. 7 |   IN | 1 |  7 || 8  | 1 | ALT0 | TxD     | 15  | 14  |
 |     |     |      0v |      |   |  9 || 10 | 1 | ALT0 | RxD     | 16  | 15  |
 |  17 |   0 | GPIO. 0 |   IN | 0 | 11 || 12 | 0 | IN   | GPIO. 1 | 1   | 18  |
 |  27 |   2 | GPIO. 2 |   IN | 0 | 13 || 14 |   |      | 0v      |     |     |
 |  22 |   3 | GPIO. 3 |   IN | 0 | 15 || 16 | 0 | IN   | GPIO. 4 | 4   | 23  |
 |     |     |    3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 | 5   | 24  |
 |  10 |  12 |    MOSI |   IN | 0 | 19 || 20 |   |      | 0v      |     |     |
 |   9 |  13 |    MISO |   IN | 0 | 21 || 22 | 0 | IN   | GPIO. 6 | 6   | 25  |
 |  11 |  14 |    SCLK |   IN | 0 | 23 || 24 | 1 | IN   | CE0     | 10  | 8   |
 |     |     |      0v |      |   | 25 || 26 | 1 | IN   | CE1     | 11  | 7   |
 |   0 |  30 |   SDA.0 |   IN | 1 | 27 || 28 | 1 | IN   | SCL.0   | 31  | 1   |
 |   5 |  21 | GPIO.21 |   IN | 1 | 29 || 30 |   |      | 0v      |     |     |
 |   6 |  22 | GPIO.22 |   IN | 1 | 31 || 32 | 0 | IN   | GPIO.26 | 26  | 12  |
 |  13 |  23 | GPIO.23 |   IN | 0 | 33 || 34 |   |      | 0v      |     |     |
 |  19 |  24 | GPIO.24 |   IN | 0 | 35 || 36 | 0 | IN   | GPIO.27 | 27  | 16  |
 |  26 |  25 | GPIO.25 |   IN | 0 | 37 || 38 | 0 | IN   | GPIO.28 | 28  | 20  |
 |     |     |      0v |      |   | 39 || 40 | 0 | IN   | GPIO.29 | 29  | 21  |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+---Pi 2---+---+------+---------+-----+-----+
```

Figure 14 - Emplacements des pins dans <wiringPi> et sur le RPI

Les paramétrages des GPIO ont étés effectués comme tel :

- **Entrées** :
  - **startProgram** : wiringPi n°27 RPI n°36
    - Correspond au bouton lançant le début du programme
      - Mise à 1 : Démarre le programme
  - **resetProgram** : wiringPi n°28 RPI n°38
    - Correspond au bouton réinitialisant le programme
      - Mise à 1 : « Reset » le programme
  - **motorReady** : wiringPi n°29 RPI n°40
    - Correspond au retour du driver sur l'état du moteur
      - Mise à 1 : Moteur prêt pour un déplacement
  - **positonMaxIn** : wiringPi n°21 RPI n°29
    - Correspond à la fin de course en position maximale de sortie
      - Mise à 0 : Position maximale de sortie atteinte
  - **positonMaxOut** : wiringPi n°22 RPI n°31
    - Correspond au la fin de course en position maximale de rentrée
      - Mise à 0 : Position maximale de rentrée atteinte
  - **positonZero** : wiringPi n°23 RPI n°33
    - Correspond au détecteur de proximité inductif donnant la position du « zéro »
      - Mise à 0 : Position « zéro » atteinte

- **Sorties** :
  - **enableMove** : wiringPi n°0 RPI n°11
    - Donne l'information au driver de son activation
      - Mise à 1 : Active le driver
  - **clockwiseMove** : wiringPi n°1 RPI n°12
    - Donne l'information au driver du choix du sens de marche du moteur
      - Mise à 1 : Sens horaire de rotation du moteur (rentrée du chariot)
  - **pulseForMove**: wiringPi n°2 RPI n°13
    - Donne l'information au driver du déplacement du moteur
      - Mise à 1 puis à 0 : Fais avancer le moteur d'un pas (1.8°), respectivement une division de pas *(1.8°/10 dans ce projet)*
  - **ledState** : wiringPi n°3 RPI n°15
    - Donne l'information à l'opérateur du mode de fonctionnement du système
      - Clignotement lent (>0.5s) : Demande de calibration
      - Mise à 1 : Mesure en cours

- Clignotement rapide (<0.2s) : Fin de la mesure

- **ledStart** : wiringPi n°4 RPI n°16
  - Donne l'information à l'opérateur de la demande d'un « start »
    - Mise à 1 : Demande d'un « start »

Le numéro de position du RPI correspond à l'emplacement physique sur celui-ci (selon le datasheet du Raspberry PI B+).

3.8.3 Insertion de la librairie <wiringPi>

Afin d'utiliser les GPIO du RPI, il faut ajouter la librairie <wiringPi>. Le nom de la commande peut se trouver sur Internet.

Il faut tout d'abord installer GIT avec la commande « `sudo apt-get install git-core` ».
Ensuite, il suffit d'envoyer la commande « `git clone git://git.drogon.net/wiringPi` » dans la console du RPI.
Puis il reste à compiler la librairie avec les commandes « `cd wiringPi` », « `git pull origin` » et « `./build` ».
Elle sera ainsi téléchargée depuis la base de données du système d'exploitation du RPI.

Pour l'utiliser dans la programmation, il n'y a plus qu'à inclure la librairie dans les classes où elle est utilisée.

3.8.4 Insertion de la librairie <libusb>

Afin d'envoyer des trames USB au capteur uPMT, il faut ajouter la librairie <libusb>. Le nom de la commande peut se trouver sur Internet.
Il suffit d'envoyer la commande « `sudo apt-get install libusb-1.0-0-dev` » dans la console du RPI.
Elle sera ainsi téléchargée depuis la base de données du système d'exploitation du RPI.

CODE: SELECT ALL

```
sudo apt-get install libusb-1.0-0-dev
```

Recherche de la librairie sur Internet

```
pi@raspberrypi ~ $ sudo apt-get install libusb-1.0-0-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  libusb-1.0-0-dev
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 184 kB of archives.
After this operation, 962 kB of additional disk space will be used.
Get:1 http://mirrordirector.raspbian.org/raspbian/ wheezy/main libusb-1.0-0-dev
armhf 2:1.0.11-1 [184 kB]
Fetched 184 kB in 10s (17.7 kB/s)
Selecting previously unselected package libusb-1.0-0-dev.
(Reading database ... 76938 files and directories currently installed.)
Unpacking libusb-1.0-0-dev (from .../libusb-1.0-0-dev_2%3a1.0.11-1_armhf.deb) ..
.
Setting up libusb-1.0-0-dev (2:1.0.11-1) ...
```

Figure 15 - Téléchargement de <libusb> sur le RPI

Pour vérifier si elle a bien été chargée sur le RPI, il faut effectuer une recherche en écrivant « `find /usr/lib -name 'libusb*'` ».

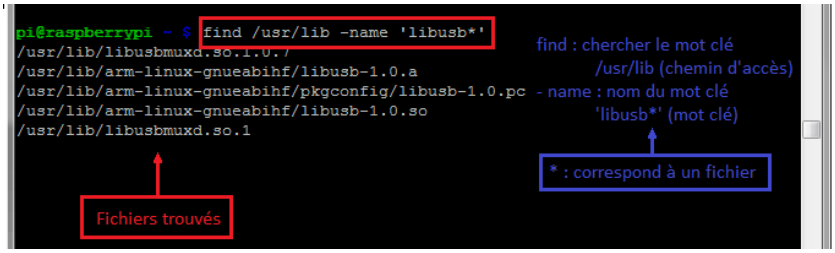


Figure 16 - Vérification du téléchargement de <libusb> sur le RPI

Tous les fichiers ajoutés apparaissent dans les résultats de la recherche.
Pour l'utiliser dans la programmation, il n'y a plus qu'à inclure la librairie dans les classes où elle est utilisée.

3.8.5 Démarrage du programme à l'enclenchement du RPI

Pour pouvoir démarrer automatiquement le programme à la mise sous tension du RPI, il faut ajouter le chemin d'accès du programme au script « rc.local ». Ce script gère les programmes qui s'exécutent une fois le RPI alimenté.

Pour ce faire, il suffit d'entrer les commandes « `cd /etc` » pour accéder au dossier « etc » et ensuite « `nano rc.local` » pour ouvrir en écriture le fichier « rc.local ».

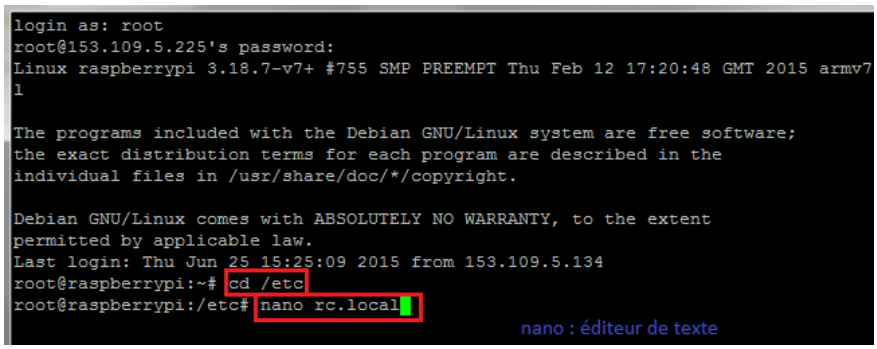


Figure 17 – Ouverture en écriture de « rc.local » dans la console du RPI

Une fois le fichier ouvert, il faut ajouter le chemin d'accès du programme qui doit être démarré. Il faut veiller à ce qu'il soit déjà compilé dans le RPI.
Si le programme contient une boucle infinie, il faut ajouter le symbole « `&` » afin que le processeur crée une boucle indépendante et qu'il continu à fonctionner en parallèle.

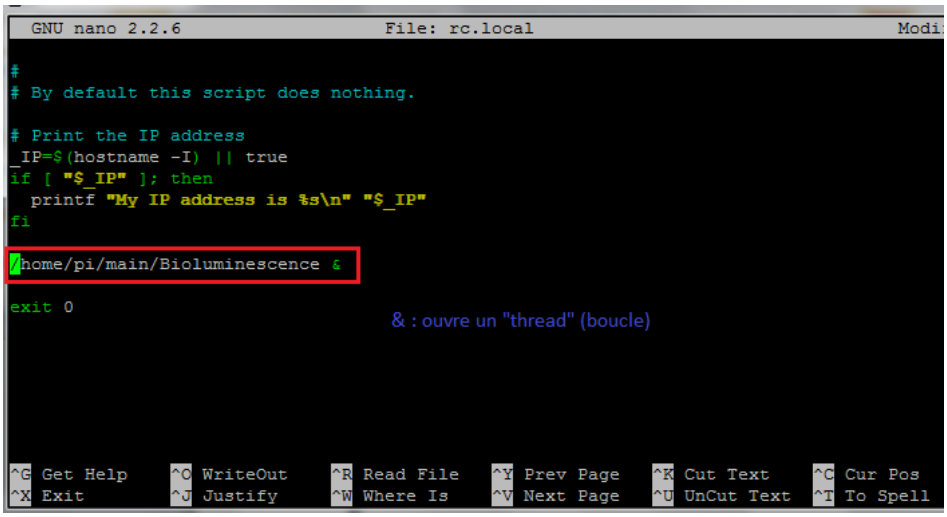


Figure 18 - Ecriture du chemin d'accès du programme qui s'exécute au démarrage du RPI dans le fichier « rc.local »

Puis il faut sauvegarder les changements avec « `ctrl X` » et « `Y` ». Au prochain démarrage et à tous les suivants, le programme choisi se lancera à la mise sous tension du RPI.

3.8.6 Sauvegarde du fichier « .csv » sur la clé USB

Pour pouvoir enregistrer le fichier de mesure sur une clé USB externe au RPI, il faut configurer un chemin d'accès où le sauver.
Il faut tout d'abord connaitre le format de la clé USB, ainsi que son chemin d'accès de base.

Pour ce faire, il suffit d'envoyer la commande « `sudo blkid` » dans la console du RPI.
Les informations des périphériques connectés vont s'afficher.

Dans ce projet une clé USB de format VFAT (Virtual FAT) est utilisée et son chemin d'accès est « dev/sda1 ». Si une seule clé USB est branchée au RPI, son emplacement sera d'office « dev/sda1 ».

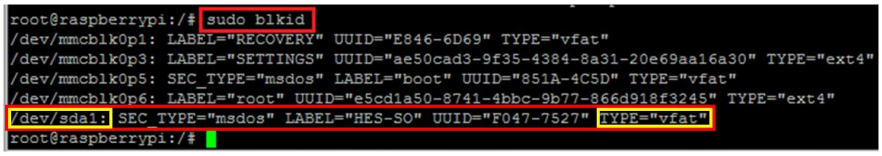


Figure 19 - Visualisation des clés USB et mémoires connectées au RPI

Une fois le format de la clé et son emplacement trouvés, il faut paramétrer, dans le fichier « fstab » du RPI, la configuration de celle-ci et l'endroit où elle pourra être lue et les fichiers sauvegardés.
Il faut entrer la commande « `sudo nano /etc/fstab` » pour ouvrir en écriture le fichier « fstab » contenu dans le dossier « etc » du RPI.
Ensuite, il suffit de rentrer les informations de la clé USB trouvées précédemment. Dans ce cas le dossier où seront enregistrées les mesures se situera dans « mnt/usb ».

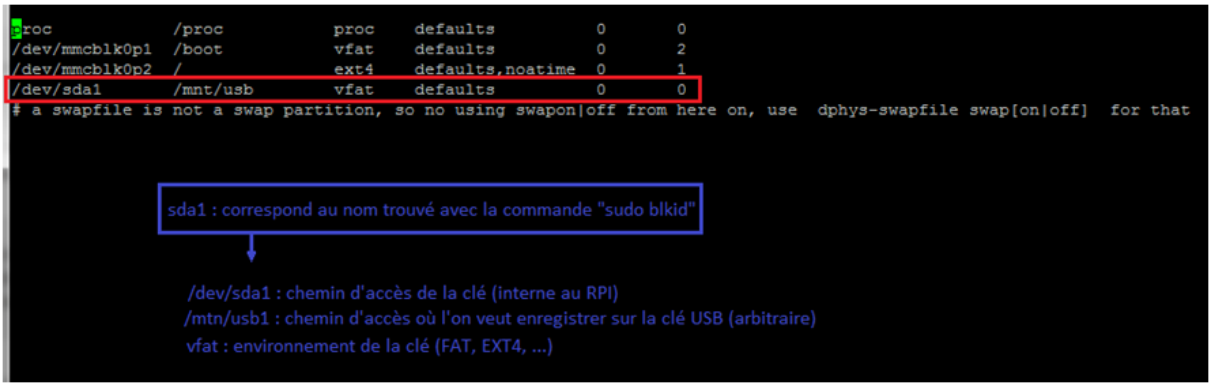


Figure 20 - Ajout du chemin d'accès de l'emplacement pour la clé USB dans le fichier « fstab »

Ensuite, il faut entrer la commande « `sudo mount -a` » pour compiler le point de montage de la clé USB et redémarrer le RPI.

Pour contrôler le bon fonctionnement, on peut accéder au nouveau dossier créé et vérifier si la clé USB est détectée. Il suffit d'employer la commande « `cd /mnt/usb` » dans la console et d'effectuer une lecture des fichiers présents avec la commande « `ls` ».

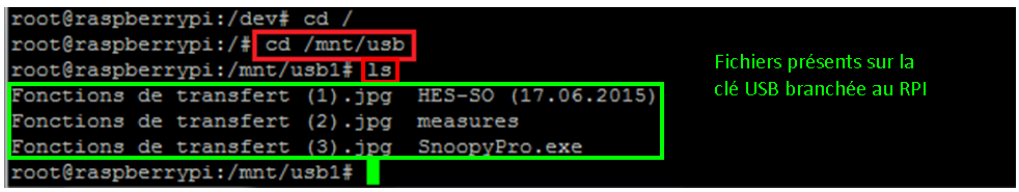


Figure 21 - Visualisation des fichiers présents sur la clé USB paramétrée

Pour ce projet, les conditions pour sauvegarder un fichier de mesure sont que la clé USB soit connectée à un port du RPI avant l'enclenchement de celui-ci et également que le format de la clé soit de type VFAT.

3.9 Mise en route

3.9.1 Séquence d'enclenchement

Pour réaliser une mesure, il faut veiller à suivre un ordre de marche.

Il faut tout d'abord insérer une clé USB de format VFAT avant l'enclenchement du RPI. Car, au démarrage de celui-ci, un script lui permet de lire si des périphériques sont présents et de leur attribuer leur propre chemin d'accès.

Figure 22 - Représentation du système de mesure avec son couvercle

Le programme se lance environ 30 secondes après que le RPI ait été allumé. La led orange sur le boitier de commande s'allume lorsque celui-ci à démarré. L'opérateur peut alors débuter la mesure en appuyant sur « Start ».

Il y a ensuite une attente de 1 seconde. Puis, la led verte se met à clignoter lentement (>0.5s). Elle indique que la séquence de calibration a commencé.
Si l'opérateur appuie sur un bouton, « → » ou « ← », qui correspondent à « Start » ou « Stop », il entre dans la phase de calibration.
Dans cette séquence il peut positionner le capteur sous le premier puits du biochip à l'aide des commandes « → » ou « ← ». Chaque impulsion sur une commande fait avancer ou reculer le capteur de 0.1mm. Il peut évidemment maintenir le bouton enfoncé pour que le déplacement s'effectue en continu.
Lorsque la position est correcte, il la laisse inchangée pendant 5 secondes et le programme sort de la phase de calibration. Il ne prendra pas le « zéro » avec le détecteur de proximité inductif lors de cette séquence de mesure. Le « zéro » étant donné par l'opérateur avec la calibration.

Si l'opérateur ne souhaite pas entrer dans la phase de calibration et laisser le système prendre le « zéro » avec le détecteur de proximité inductif, il ne touche rien pendant les 5 secondes où la led verte clignote lentement (>0.5s).

Dans le deux cas, le programme attend un nouveau « Start » pour commencer la mesure. La led orange l'indique à l'opérateur.

Une fois le bouton poussé, le programme se met soit en position « zéro » avec le détecteur de proximité inductif, ou reste à la position qui a été calibrée par l'opérateur.

Il commence ensuite la mesure à proprement dit. Le capteur uPMT va se déplacer sous chaque puits du biochip pendant dix secondes. Une fois arrivé au dernier puits, il revient en position de départ et recommence le cycle ainsi de suite, pendant 2 heures. Pendant toute la phase de mesure, la led verte reste allumée en permanence.

Lorsque la séquence de mesure est terminée, elle se met à clignoter rapidement (<0.2s) et la led orange s'allume pour avertir l'opérateur que la clé USB peut être retirée et qu'un nouveau « Start » est désiré, si il veut relancer le programme depuis le début.

En tout temps le bouton « Stop » peut être activé et le système s'arrête et se remet au début du programme, sauf si pendant la calibration où le « Stop » permet de remplacer le bouton « ← » et permet de faire avancer le capteur.

Toutefois, lorsque le capteur prend une mesure, c'est-à-dire pendant les dix secondes où il est à l'arrêt sous le puits, le programme n'accepte pas le « Stop » car il doit respecter un ordre d'enclenchement et de déclenchement pour arrêter le capteur.

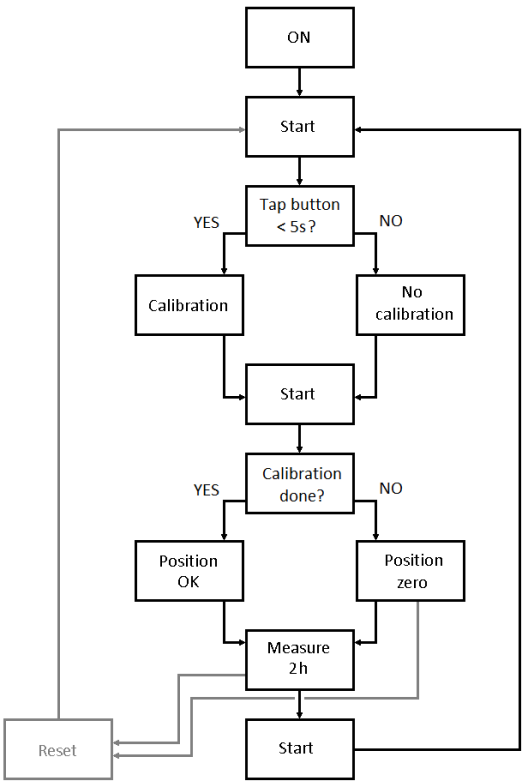


Figure 23 - Séquence de mesure, une led orange indique la demande d'un « Start » et une led verte indique la demande de calibration, si la mesure est en cours ou finie

4. Résultats

4.1 Mesure de la consommation électrique moyenne

La consommation électrique du système de mesure complet comprend l'alimentation du moteur ainsi que la consommation du RPI, du driver et l'allumage des leds. Il faut aussi tenir compte de l'alimentation du relais et le détecteur de proximité inductif.

Le pire des cas survient lorsque le moteur se déplace. A ce moment, le moteur est alimenté, mais également le RPI, le driver et la led verte.

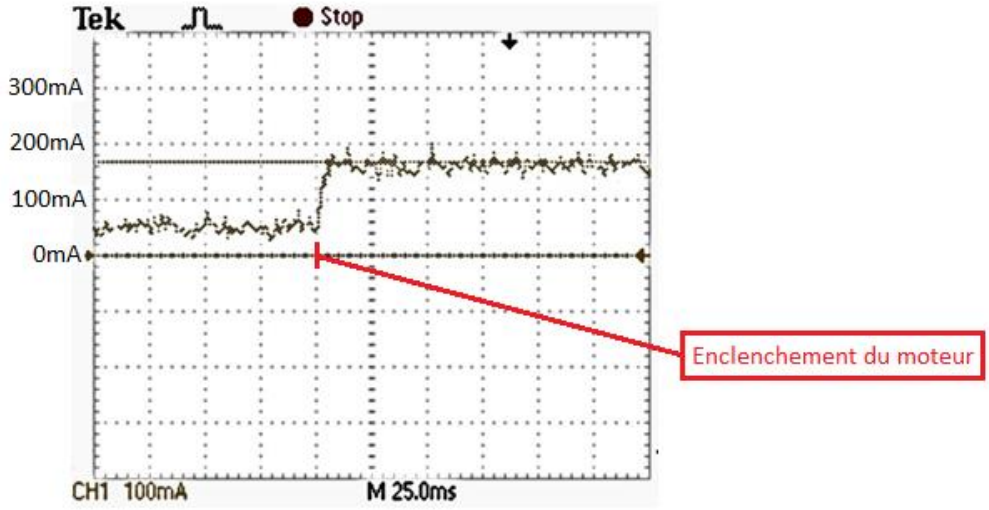


Figure 24 - Courbe du courant mesurée sur les bornes d'alimentation du système après le démarrage du moteur

L'intensité est mesurée directement sur le 24V.
Il y a environ 60mA lorsque le moteur est à l'arrêt et aux alentours de 170mA lorsque celui-ci tourne.
Il est activé 60 secondes pour une série de mesures qui dure aux alentours de 180 secondes. Donc le moteur tourne environ un tiers du temps.

Pour connaître la puissance moyenne consommée par le système, il faut appliquer la formule suivante :

$$P_{moyenne} = U_{batterie} \times \frac{(I_{M,IN} + 2 \times I_{M,OUT})}{3} = 24 \times \frac{(0.17 + 2 \times 0.06)}{3} = 2.32W$$

, avec $U_{batterie}$: Tension moyenne délivrée par la batterie [V]
$I_{M,IN}$: Courant mesurée lorsque le moteur tourne [A]
$I_{M,OUT}$: Courant mesurée lorsque le moteur ne tourne pas [A]

4.2 Mesure de la répétabilité de positionnement

Deux mesures de distance sont effectuées. Elles sont prises entre la flasque du moteur PAP et le bord du chariot mobile.

La première mesure est faite après que le système ait pris le « zéro » et se soit arrêté dans la position avant le premier puits.
La deuxième est prise à la fin des 2 heures d'expérience, lorsque le chariot revient à cette même position d'arrêt.

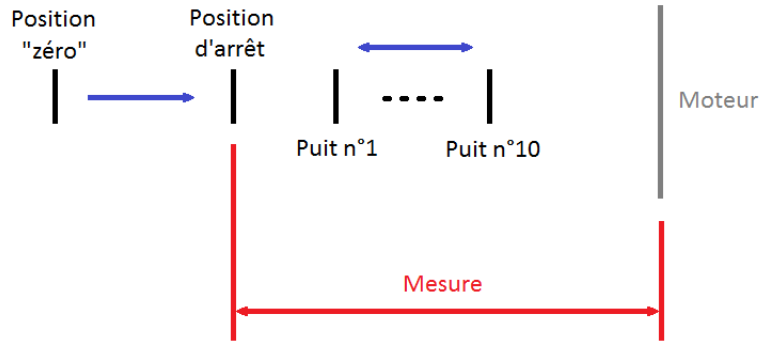


Figure 25 - Schéma de la prise de mesure de distance entre le bord du chariot en position d'arrêt et la flasque du moteur

Les valeurs mesurées valent :

Temps « t » depuis le début de l'expérience	Nombre de séries (10 puits) de mesures de bioluminescence	Distance entre le moteur et le chariot, en position d'arrêt [mm]
0	0	62.10
2h	45	62.00

Tableau 2 - Valeurs des mesures de la répétabilité de positionnement

Le glissement moyen entre une série de mesures et la suivante vaut :

$$Glissement = \frac{d_{début} - d_{finale}}{N_{séries}} = \frac{62.10 - 62.00}{45} = 0.0022\ mm$$

, avec d_{finale} : Distance entre le support du moteur et le chariot à la dernière mesure [mm]
$d_{début}$: Distance entre le support du moteur et le chariot à la première mesure [mm]
$N_{séries}$: Nombre de séries de mesures lors de l'expérience

4.3 Mesure dans l'obscurité (sans biochip)

Lorsque le capteur de photons, même sans être alimenté, est en présence de forte lumière, *par exemple celle d'une lampe ou d'un tube fluorescent*, il ne doit pas être utilisé pour une mesure pendant une trentaine de minutes.
Ceci s'explique par le fait qu'il garde une « empreinte » de la lumière qui a frappée la tête de mesure.

Si une mesure est effectuée durant ce laps de temps, les premières valeurs seront erronées.

Pour cette expérience, le capteur a été exposé à la lumière d'un tube fluorescent de 36W. Il a été placé à 2 mètres de celui-ci, pendant 5 minutes.
Puis, il a été plongé dans l'obscurité pendant un certain temps.
Pour finir une série de mesures de 2 heures, sans biochip, a été prise.

Tous les tableaux contenant les valeurs numériques des mesures des graphiques suivants sont mis en annexe. ***[Annexes : 7.1.1 à 7.1.4]***

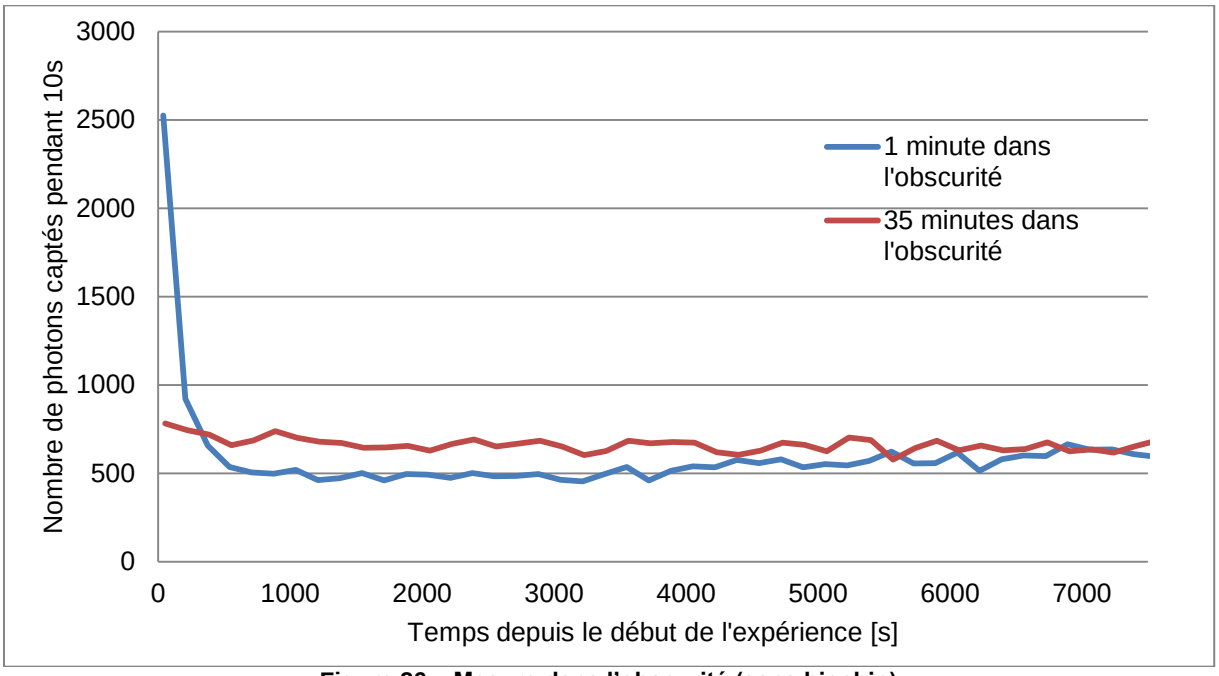


Figure 26 – Mesure dans l'obscurité (sans biochip)

Bleu : 1 minute dans l'obscurité avant le début de l'expérience

Rouge : 35 minutes dans l'obscurité avant le début de l'expérience

Variabilité des mesures pour les dix puits (après 1 minute d'obscurité) à t = 1'000s :
moyenne = 409photons/10s et écart type = 44photons/10s

Variabilité des mesures pour les dix puits (après 1 minute d'obscurité) à t = 4'000s :
moyenne = 477photons/10s et écart type = 31photons/10s

Variabilité des mesures pour les dix puits (après 35 minutes d'obscurité) à t = 1'000s :
moyenne = 670photons/10s et écart type = 21photons/10s

Variabilité des mesures pour les dix puits (après 35 minutes d'obscurité) à t = 4'000s :
moyenne = 641photons/10s et écart type = 43photons/10s

5. Discussions

5.1 Mesure de la consommation électrique moyenne

La consommation moyenne trouvée est de 2.32W. Le système fonctionnant 2heures par jours, l'énergie moyenne consommée par jour est de 4.64Wh et 139.2Wh pendant un mois.

Le système étant embarqué sur des bouées, l'alimentation est fournie par des batteries de $24V_{dc}$. Elles sont rechargées par des panneaux solaires.

Avec une si faible consommation journalière, le système de mesure ne viderait pas totalement les batteries, même en absence de soleil prolongée. En effet, une batterie standard, qui peut fournir 1'200Ah, suffirait à garantir l'alimentation de cette partie du projet.

L'énergie que peut fournir une telle batterie se calcule à l'aide de la formule suivante :

$$E_{batterie} = U_{batterie} \times C_{batterie} = 24 \times 1'200 = 28'800Wh$$

, avec $U_{batterie}$: Tension aux bornes de la batterie [V]
$C_{batterie}$: Charge de la batterie [Ah]

5.2 Mesure de la répétabilité de positionnement

Après les 45 allers-retours pendant la mesure, le glissement total est de 0.1mm.

La principale raison de cet écart est que, lors de la première mesure de distance, le moteur se déplace depuis la position « zéro » jusqu'au biochip, donc dans le sens avant, alors que lors de la deuxième mesure, après 2 heures, le chariot se déplace depuis la position du dixième puits jusqu'à la position d'arrêt du premier puits, donc dans le sens de recule.

Il y'a donc l'effet du jeu du chariot et de la tige filetée qui rentre en ligne de compte pour expliquer cet écart.

Cette valeur de 0.1mm rentre dans la tolérance de positionnement acceptée pour que la tête de mesure se situe toujours sous la surface des puits.

5.3 Mesure dans l'obscurité (sans biochip)

Lors d'une mesure avec un vrai biochip, les valeurs attendues se situent autour de 30'000 photons pour dix secondes.

L'influence de « l'inertie » lumineuse est donnée par la formule suivante :

$$Erreur_{inertie} = \frac{(Inertie_{maximale} + Valeur_{attendue}) - (Inertie_{minimale} + Valeur_{attendue})}{Valeur_{attendue}}$$

, avec $Inertie_{maximale}$: Valeur maximale de « l'inertie » mesurée lors d'un test dans l'obscurité [photons/10s]
$Inertie_{minimale}$: Valeur minimale de « l'inertie » mesurée lors d'un test dans l'obscurité [photons/10s]
$Valeur_{attendue}$: Valeur moyenne mesurée lors d'un test avec des bactéries [photons/10s]

On voit que le graphique, qui montre l'évolution de la courbe au cours du temps, après seulement 1 minute dans l'obscurité avant le début de la mesure, met environ 15 minutes à se stabiliser.
Le nombre de photons moyen pour les dix puits, lors de la première mesure, se situe autour de 2'500.
Alors que le nombre de photons moyen pendant dix secondes, après 15 minutes, est plutôt de 500.

Lors d'une véritable mesure avec un biochip, on ne pourrait pas récolter de valeurs correctes pendant les 15 première minutes, si le capteur de photons à subit une exposition à la lumière.

$$Erreur_{inertie} = \frac{(2'500 + 30'000) - (500 + 30'000)}{30'000} = 6.67\ \%$$

La première mesure comporte une erreur jusqu'à 6.67%, par rapport aux valeurs attendues.
Il est donc préférable de ne pas lancer une série de mesures, pour ne pas enregistrer de valeurs erronées.

Alors que la courbe, pour laquelle le capteur de photons est resté 35 minutes dans l'obscurité avant le début de la prise de mesure, permet d'avoir des mesures beaucoup plus correctes.
En effet, la première mesure vaut environ 700 photons pendant dix secondes et se stabilise autour de 600.

L'erreur due à « l'inertie » lumineuse est négligeable après une trentaine de minutes dans l'obscurité.

$$Erreur_{inertie} = \frac{(700 + 30'000) - (600 + 30'000)}{30'000} = 0.33\%$$

La première mesure comporte une erreur jusqu'à 0.33%, par rapport aux valeurs attendues.
Cette erreur est négligeable, car par la suite des approximations seront faites pour trouver la concentration de polluant et, en dessous de 1% d'écart sur la mesure, ces approximations restent valables.

Donc pour lancer une mesure si le capteur de photons a été exposé à la lumière, il faut attendre une période de 30 minutes pour assurer des valeurs correctes.

6. Conclusion & Améliorations

6.1 Conclusion

Selon le cahier des charges, qui est inscrit dans la donnée du travail de diplôme, le développement et la construction du système de mesure de bioluminescence a été réalisé correctement.
Par contre, la fermeture hermétique du biochip a été mandatée à une entreprise extérieure, car nous ne disposons pas du matériel pour réaliser ces travaux.
L'analyse et les calculs des concentrations des polluants n'ont pas pu être faits. Il manquait du temps et des connaissances sur ce sujet. Mais une fois l'équation, qui relie les mesures et la concentration de pollution connues, cette formule n'a plus qu'à être entrée dans le code de programmation du RPI.

Au terme du travail, le prototype est terminé. Le système permet d'effectuer des mesures de bioluminescence sur 2 heures et de récolter les données sous forme de tableau Excel.

Quelques modifications ont été apportées au cahier des charges durant le projet.
L'acquisition des données se fait sur une clé USB externe au boitier. Les commandes de démarrage et ainsi que d'arrêt se donnent à l'aide d'un bouton poussoir à trois positions.

Des mesures ont été faites, à l'UNIL de Lausanne, mais elles se sont avérées inexactes et invalidées. L'explication est que les bactéries, génétiquement modifiées, étaient en phase de décroissance et elles n'ont pas réussies à jouer leur rôle de bio-émetteur.
Une nouvelle série de mesures sera effectuée pendant le mois d'août 2015, au département de microbiologie fondamentale de l'UNIL. Ces résultats seront intégrés à la présentation orale qui se tiendra en septembre 2015*(date à confirmer)* à la HES-SO Valais/Wallis de Sion.

6.2 Améliorations & Travaux futurs

La première série d'améliorations concerne le système de mesure.

Il faut en effet remplacer les commandes « Start » et « Stop » pour qu'elles puissent être envoyées depuis l'ordinateur central. Le protocole de communication est laissé libre.

Il faut également que les données mesurées et enregistrées sous un tableau au format « .csv » soient transmises à ce même ordinateur central. Le traitement des données peut se faire directement par le RPI et ainsi il n'enverrait plus que la valeur de concentration de polluant par puits. Dans ce cas, il faut que le RPI puisse stocker les fichiers de mesures pour qu'ils puissent être transmis si besoin.

Ce projet est un prototype, des composants peuvent être changés.
Le driver du moteur PAP peut être remplacé par une électronique plus simple et moins encombrante. En effet, il est assez complet, car il est conçu par SONCEBOZ pour être utiliser avec tous les moteur PAP de 0.2A/phase à 3.75A/phase et peut effectuer des divisons de pas jusqu'à 40. Alors que dans ce projet seul une division de 10 est utilisée et le moteur consomme uniquement 0.34A/phase.

Le moteur PAP avec son système d'entrainement linéaire peut être réduit, car la course maximale est de 100mm alors que seulement 40mm sont nécessaire.

Comme présenté dans les mesures, si le capteur de photons est exposé à la lumière, il doit être placé au moins 30 minutes dans l'obscurité. Il faudrait prévoir un cache qui assurerait son étanchéité lumineuse lorsqu'un biochip est inséré sur le support.
Autrement, il faut ajouter dans le code du programme une attente de 30 minutes avant la prise de mesures.

La deuxième série d'améliorations consiste à intégrer le système de mesure dans le projet global. C'est-à-dire, concevoir la partie qui s'occupe du remplissage du biochip ainsi que la partie qui gère les déplacements de biochip aux différentes positions.

Le dispositif de remplissage comprendra une à deux pompes péristaltiques pour obtenir un échantillon d'eau de mer et rejeter le surplus. Ensuite un système devra injecter une quantité similaire de liquide dans tous les puits.

Le déplacement du biochip peut s'effectuer de deux manières, comme expliqué comme suit :

- Les trente biochips sont placées sur un magasin rotatif, qu'amène chaque mois l'opérateur. Ce magasin, en forme de disque, se déplace lors d'une demande de mesure. Il s'arrête à une première position pour que le biochip puisse être remplis d'eau de mer et à une deuxième position pour qu'il puisse subir la séquence de mesure.

- Une boite contient les trente biochips. Lors d'une demande de mesure, un piston pousse le biochip, situé au fond de cette boite, aux deux positions d'arrêt, celle de remplissage d'eau et celle de mesure. Puis le piston fini sa course en envoyant le biochip dans une seconde boite, qui fait office de « stock » de déchets. Il revient ensuite à la position initiale et attend une nouvelle demande de l'ordinateur central.

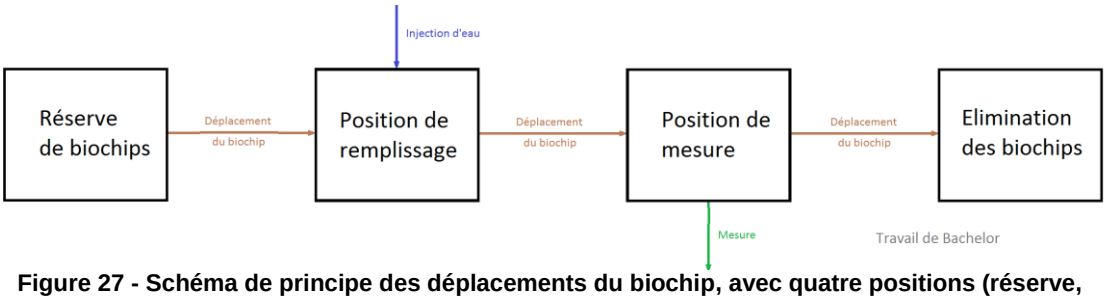


Figure 27 - Schéma de principe des déplacements du biochip, avec quatre positions (réserve, remplissage, mesure et élimination)

Une fois les différentes parties fabriquées, il reste à intégrer le tout dans la bouée.

A la fin, celle-ci comprendra les éléments suivants :

- **Alimentation électrique** : Batteries et panneaux solaires
- **Remplissage du biochip** : Pompes et dispositif d'injection de l'eau

- **Déplacements du biochip** : Magasin rotatif ou boite de réserve avec système d'entrainement linéaire qui s'arrête aux différentes positions et élimine le chip à la fin du cycle
- **Système de mesure** (conçu dans ce projet) : Capteur de photons solidaire d'un chariot déplacé par un moteur PAP, le tout commandé par un micro-ordinateur qui acquisitionne et transmet les valeurs mesurées

Le tout étant commandé par un ordinateur central situé sur les côtes.

7. Références

7.1 Publications

[1.1] – *BAUMER – Septembre 2006 – N° 802297 – Détecteurs de proximité inductifs et capacitifs (Catalogue) – P.1.14*

7.2 Images

[2.1] – *Raspberry Pi B+ – www.raspberrypi.org*
[2.2] – *Datasheet – SMD 9103 – SONCEBOZ*
[2.3] – *Datasheet – uPMT Evaluation Kit – HAMAMATSU – Juin 2013*
[2.4] – *GPIO disposition – www.rs-online.com*

8. Annexes

[1] – **Schémas électriques**
- [1.1] – Commande 24V
- [1.2] – Commande 3.3V

[2] – **Code de programmation**
- [2.1] – Bioluminescence.cpp
- [2.2] – Motor.cpp
- [2.3] – Motor.h
- [2.4] – USBTransfer.cpp
- [2.5] – USBTranfer.h

[3] – **Schémas mécaniques**
- [3.1] – Assemblage
  - [3.1.1] – Système complet
  - [3.1.2] – Système de mesure
- [3.2] – Châssis
  - [3.2.1] – Perçages latéraux
  - [3.2.2] – Perçages M2 et M3
  - [3.2.3] – Perçages M4
  - [3.2.4] – Trous de passage
- [3.3] – Plots
  - [3.3.1] – Moteur EST
  - [3.3.2] – Moteur NORD
  - [3.3.3] – Moteur SUD
- [3.4] – Supports
  - [3.4.1] – Fin de course
  - [3.4.2] – Capteur socle
  - [3.4.3] – Capteur latéral
  - [3.4.4] – Capteur médian
  - [3.4.5] – Capteur renvoi
- [3.5] – Accessoires
  - [3.5.1] – Paroi
  - [3.5.2] – Biochip

[4] – **Impression 3D**
[4.1] – Pièces imprimées en 3D

[5] – **Datasheets**
[5.1] – Raspberry Pi B+ : Installation manual
[5.2] – Driver : SMD 9103 SONCEBOZ
[5.3] – Moteur PAP : 8453 SONCEBOZ
[5.4] – Capteur de photons : uPMT HAMAMATSU
[5.5] – Evaluation kit : uPMT Evaluation Kit HAMAMATSU
[5.6] – Détecteur de proximité inductif : IFRM 08P1701 L BAUMER
[5.7] – Relais 24V : 40.61 FINDER
[5.8] – Convertisseur DC/DC : TEN 10-2411 TRACO

[6] – **Protocole**
[6.1] – Evaluation kit (uPMT EK1) : Protocole

[7] – **Mesures**
[7.1] - Mesure dans l'obscurité (sans biochip)
[7.1.1] – Tableau des mesures après 1 minute dans l'obscurité
[7.1.2] – Tableau des mesures après 15 minutes dans l'obscurité
[7.1.3] – Tableau des mesures après 35 minutes dans l'obscurité
[7.1.4] – Tableau des mesures après 55 minutes dans l'obscurité

9. Remerciements

- HES-SO Valais/Wallis :
    - Prof. Martial Geiser, pour ses conseils relatifs au projet et son organisation
    - Prof. Samuel Chevailler, pour ses conseils concernant le choix du moteur
    - M. Frédéric Truffer, pour sa collaboration durant tout le projet
    - M. Serge Amoos, pour ses conseils relatifs à la conception mécanique
    - M. Olivier Walpen, pour son aide concernant les pièces découpées au laser
    - M. Pascal Sartoretti, pour son aide pour les impressions 3D des pièces

- UNIL, Department of Fundamental Microbiology :
    - Prof. Jan Roelof Van Der Meer, pour sa collaboration
    - Dr. Yoann Le Digabel, pour la préparation des échantillons de mesure

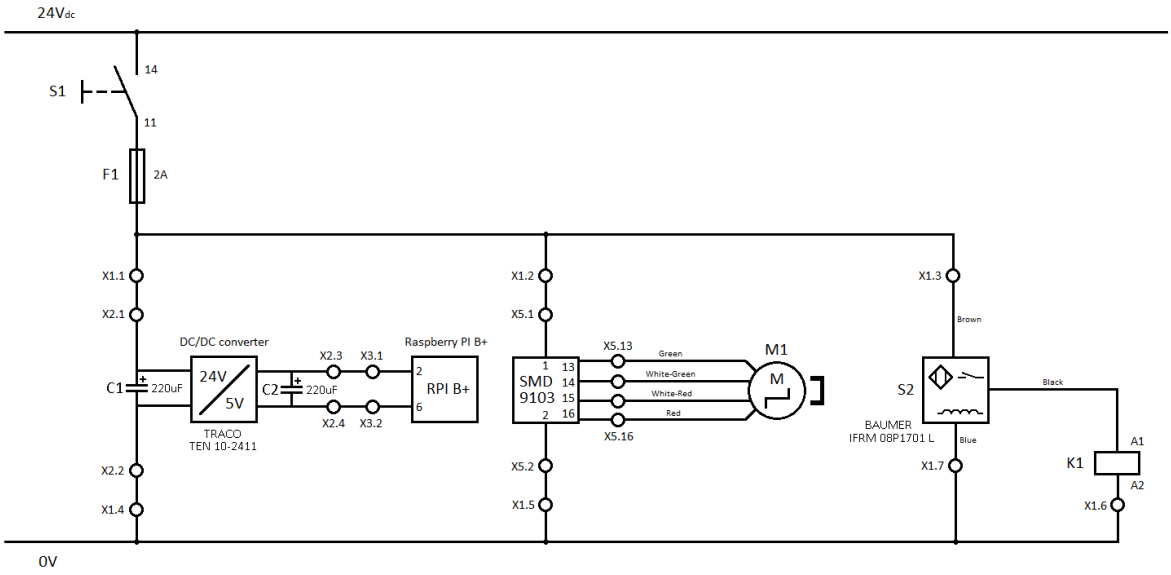
Partenariat

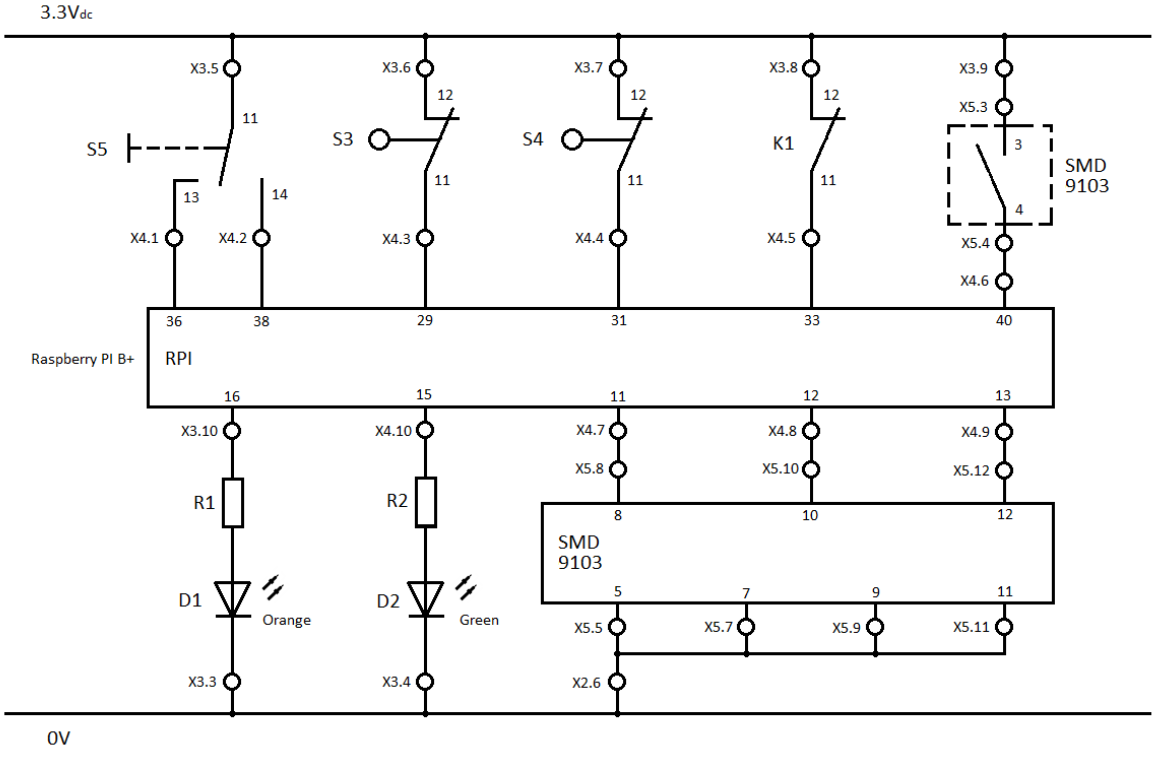
UNIL | Université de Lausanne

Department of Fundamental Microbiology
UNIL | Université de Lausanne
Lausanne, Suisse

Signature

Vincent Giachino


24Vdc
S1
14
11
F1
2A
X1.1
X2.1
X2.2
X1.4
DC/DC converter
24V
5V
C1
220uF
C2
220uF
TRACO
TEN 10-2411
X2.3
X3.1
X2.4
X3.2
Raspberry PI B+
RPI B+
X1.2
X5.1
X5.2
X1.5
SMD
9103
X5.13
X5.16
Green
White-Green
White-Red
Red
M1
M
X1.3
Brown
S2
Black
Blue
BAUMER
IFRM 08P1701 L
X1.7
K1
A1
A2
X1.6
0V


3.3Vdc
X3.5
S5
11
13
14
X4.1
X4.2
X3.6
S3
12
11
X4.3
X3.7
S4
12
11
X4.4
X3.8
K1
12
11
X4.5
X3.9
X5.3
3
4
SMD
9103
X5.4
X4.6
36
38
29
31
33
40
Raspberry PI B+
RPI
16
15
11
12
13
X3.10
X4.10
X4.7
X4.8
X4.9
X5.8
X5.10
X5.12
R1
R2
8
10
12
SMD
9103
D1
Orange
D2
Green
5
7
9
11
X5.5
X5.7
X5.9
X5.11
X3.3
X3.4
X2.6
0V

```
//============================================================================
// Name        : Bioluminescence.cpp
// Author      : Vincent Giachino
// Version     : 1.0
// Copyright   : HES-SO Valais/Wallis
// Description : This is the program for the part
//                               of measurement of the BRAAVOO project
//============================================================================

#include "Motor.h"
#include "USBTransfer.h"
#include <time.h>
#include <iostream>
#include <sstream>
#include <fstream>
#include <string>
#include <wiringPi.h>

using namespace std;

#define ON          1
#define OFF         0

/***Variables declaration***/

/*Conditions variables*/
bool startExperimentation = false; //Check if the experimentation can be started
bool calibrationRequest = false; //Check if a calibration is requested
bool calibrationDone = false; //Check if the calibration is done
bool calibrationPerformed = false; //Check if a calibration is performed
bool calibrationError = false; //Check if an error appear while the calibration
bool fileCSVIsOK = false; //Check if the file CSV is created and initialized
bool sensorIsOK = false; //Check if the sensor is detected and initialized
bool leaveSequence = false; //Check if the program must be left

/*Time and period variables*/
int periodForMessage = 1; //Period for send the message to the console (in s)
int periodOfBlinkingLed = 500; //Period for the flashing Led work (in ms)
int timeForRequestCalibration = 5000; //Time for request a calibration at the beginning (in ms)
int timeActualForCalibration = 0; //Actual time for the request of calibration (in ms)
int timeForOneSerialOfMeasure = 160; //Time for one serial of measure (in seconds)
int timeOfExperimentation = 120; //Time of the experimentation (in minutes)

/*Mechanical variables*/
int numberOfHoles = 10; //Number of holes in the biochip
double minimumStepForCalibration = 0.1; //Distance step for the calibration (in mm)
double distanceBetweenBiochipAndZero = 10.6; //Distance between the biochip and the position zero
(in mm)
double distanceBetweenTwoHole = 2.55; //Distance between two hole of the biochip (in mm)
int numberOfPulse; //Number of pulse between two hole of the biochip
double correctionFactor = -0.1; //Correction factor of distance for the return at the 1st position
(in mm)

/*Receive data declaration*/
int headerLength = 4; //Size oh the header
int dataLength = 64; //Size of the receive data
int tempData = 0; //Temporal value for save the received data
time_t timeBegin; //Time at the beginning of the measure sequences
time_t timeMeasure; ///Time at each measure
double differenceTime = 0; //Difference between each measure

/*File CSV declaration*/
ostringstream fileName;
string timeFile;
```

```
struct tm * timeInfo;
int tempName = 0;
int tempOldName = 0;

int main() {

	/***Begin of the main program***/

	/*Infinite loop*/
	while (true) {

		/*Set the name for the file CSV*/
		if (tempName <= tempOldName) {

			tempName = tempOldName + 1;
			tempOldName = tempName;

		}

		time(&timeBegin);
		timeInfo = localtime(&timeBegin);
		timeFile = asctime(timeInfo);

		fileName.str("");
		fileName << "/mnt/usb/Measure_" << tempName << "_" << timeInfo->tm_hour
				<< "h" << timeInfo->tm_min << "m" << timeInfo->tm_sec << "s"
				<< "_" << timeInfo->tm_mday << "d" << (1 + timeInfo->tm_mon)
				<< "m" << (1900 + timeInfo->tm_year) << "y" << ".csv";

		/*Constructor classes declaration*/
		Motor myMotor;
		USBTransfer sensor;

		/*Start request*/
		cout << "Press <<Start>>" << endl;
		sleep(periodForMessage);

		myMotor.turnOnStartLed();

		while (true) {
			if (myMotor.startEnable()) {
				break;
			}
		}

		myMotor.turnOffStartLed();

		/*Initialize classes*/
		if (sensor.initUSB()) {

			sensorIsOK = true;
			cout << "Sensor is initialized" << endl;
			sleep(periodForMessage);

		} else {

			sensorIsOK = false;
			myMotor.turnOffStateLed();

			cout << "Sensor is not detected" << endl;
			sleep(periodForMessage);

		}
```

```
ofstream fileCSV(fileName.str().c_str());

if (fileCSV) {

	fileCSVIsOK = true;
	cout << "File CSV is created" << endl;
	sleep(periodForMessage);

} else {

	fileCSVIsOK = false;
	myMotor.turnOffStateLed();

	cout << "File CSV is not created" << endl;
	sleep(periodForMessage);

}

/*Formating the file CSV*/
fileCSV << "Project BRAAVOO" << ";";
fileCSV << "Bioluminescence measure" << endl;
fileCSV << "Measurement sequence start at (GMT : +0) : " << timeFile
		<< endl;

for (int i = 0; i < numberOfHoles; i++) {

	fileCSV << "Time [s]" << ";";
	fileCSV << "Hole n° " << (i + 1) << ";";

}

fileCSV << endl;

/***Begin of the program of the experimentation***/

/*Start if the sensor is detected and initialized*/
while (sensorIsOK && fileCSVIsOK) {

	/*Reset of the variables*/
	startExperimentation = false;
	calibrationRequest = false;
	calibrationDone = false;
	calibrationPerformed = false;
	calibrationError = false;
	leaveSequence = false;
	timeActualForCalibration = 0;

	/***Calibration sequences***/

	/*Calibration request*/
	cout << "Tap any button to request a calibration" << endl;
	sleep(periodForMessage);

	while (timeActualForCalibration <= timeForRequestCalibration) {

		/*Blinking the led*/
		myMotor.turnOnStateLed(periodOfBlinkingLed / 2, ON);

		if (myMotor.startEnable() || myMotor.resetEnable()) {

			cout << "Calibration is requested" << endl;
			sleep(periodForMessage);

			calibrationRequest = true;
```

```
                break;

        }

        timeActualForCalibration += (int) (periodOfBlinkingLed / 2);

    }

    /*Calibration in progress*/
    while (calibrationRequest) {

        if (myMotor.startEnable()) {

            if (!myMotor.move(minimumStepForCalibration, BACK, ON)) {

                calibrationError = true;
                break;

            }

        }

        if (myMotor.resetEnable()) {

            if (!myMotor.move(minimumStepForCalibration, FORWARD,
            ON)) {

                calibrationError = true;
                break;

            }
        }

        for (int i = 0; i < timeForRequestCalibration; i++) {

            if (myMotor.startEnable() || myMotor.resetEnable()) {
                break;
            }

            if (i >= (timeForRequestCalibration - 1)) {

                calibrationDone = true;
                break;

            }

            delay(1);

        }

        /*Calibration performed*/
        if (calibrationDone) {

            calibrationRequest = false;
            calibrationDone = false;
            calibrationPerformed = true;

            if (!calibrationError) {
                {

                    cout << "Calibration is done" << endl;
                    sleep(periodForMessage);

                }
```

```
                    break;

                }

            }

        }

        /*Calibration with an error*/
        if (calibrationError) {

            break;

        }

        /***Measurement sequences***/

        /*Start request*/
        cout << "Press <<Start>>" << endl;
        sleep(periodForMessage);

        myMotor.turnOnStartLed();

        while (true) {
            if (myMotor.startEnable()) {
                break;
            }
        }

        myMotor.turnOffStartLed();

        while (true) {

            if (myMotor.startEnable()) {

                startExperimentation = true;
                break;

            }

            if (myMotor.resetEnable()) {

                startExperimentation = false;
                break;

            }

        }

        /*Movement in progress*/
        while (startExperimentation) {

            /*Start the time for the file CSV*/
            time(&timeBegin);

            cout << "Program of measurement is started" << endl;
            sleep(periodForMessage);

            /*Turn on the led*/
            myMotor.turnOnStateLed(periodOfBlinkingLed, OFF);

            /*Positioning before the 1st hole*/
            if (!calibrationPerformed) {
```

```
		if (myMotor.movePositionZero()) {

			cout << "Position zero is done" << endl;
			sleep(periodForMessage);

		} else {

			leaveSequence = true;
			break;

		}

		if (myMotor.move(distanceBetweenBiochipAndZero,
		FORWARD, OFF)) {

			cout << "Position before the 1st hole is done" << endl;
			sleep(periodForMessage);

		} else {

			leaveSequence = true;
			break;

		}

	} else {

		if (myMotor.move(distanceBetweenTwoHole, BACK,
		OFF)) {

			cout << "Position before the 1st hole is done" << endl;
			sleep(periodForMessage);

		} else {

			leaveSequence = true;
			break;

		}

	}

	/*Positioning below all holes and measuring sequence*/
	for (int a = 0;
			a <= int((timeOfExperimentation * 60)
					/timeForOneSerialOfMeasure); a++) {

		for (int i = 0; i < (numberOfHoles / 2); i++) {

			if (myMotor.move(distanceBetweenTwoHole, FORWARD,
			OFF)) {

				cout << "Ready for the measure below the hole n° : "
						<< (i + 1) << endl;

				tempData = sensor.sequenceOfTransfer(sensor,
						headerLength, dataLength);

				time(&timeMeasure);
				differenceTime = difftime(timeMeasure, timeBegin);

				fileCSV << (int) differenceTime << ";";
				fileCSV << tempData << ";";
```

```
                cout << "There is  " << tempData
                      << " photons emitted while 10s on the hole n° : "
                            << (i + 1) << endl;

          } else {

                leaveSequence = true;
                break;

          }

    }

    if (leaveSequence) {
          break;
    }

    if (myMotor.move(distanceBetweenTwoHole, FORWARD,
    OFF)) {

          cout << "Position before the 6th hole is done" << endl;
          sleep(periodForMessage);

    } else {

          leaveSequence = true;
          break;
    }

    for (int i = 0; i < (numberOfHoles / 2); i++) {

          if (myMotor.move(distanceBetweenTwoHole, FORWARD,
          OFF)) {

                cout << "Ready for the measure below the hole n° : "
                            << (i + 6) << endl;

                tempData = sensor.sequenceOfTransfer(sensor,
                            headerLength, dataLength);

                time(&timeMeasure);
                differenceTime = difftime(timeMeasure, timeBegin);

                fileCSV << (int) differenceTime << ";";
                fileCSV << tempData << ";";

                cout << "There is  " << tempData
                      << " photons emitted while 10s on the hole n° : "
                            << (i + 6) << endl;

          } else {

                leaveSequence = true;
                break;

          }

    }

    if (leaveSequence) {
          break;
    }
```

```
                        /*Back before the 1st hole*/
                        cout << "Serial measure n° " << (a + 1) << " is done"
                                    << endl;
                        fileCSV << endl;

                        numberOfPulse = (distanceBetweenTwoHole
                                    * (numberOfHoles + 1) - correctionFactor)
                                    / myMotor.advancePerPulse;

                        if (myMotor.move(numberOfPulse, BACK)) {

                              cout << "Position before the 1st hole is done" << endl;
                              sleep(periodForMessage);

                        } else {

                              leaveSequence = true;
                              break;

                        }

                  }

                  cout << "The measurement is finished" << endl;
                  sleep(periodForMessage);

                  /*Close the file CSV*/
                  fileCSV.close();

                  if (fileCSVIsOK) {
                        cout << "The file CSV is saved on the USB key" << endl;
                  }

                  /*Reset of the variables*/
                  startExperimentation = false;
                  calibrationRequest = false;
                  calibrationDone = false;
                  calibrationPerformed = false;
                  calibrationError = false;
                  leaveSequence = false;
                  timeActualForCalibration = 0;

                  /*Turn off the state led*/
                  myMotor.turnOffStateLed();

            }

            /*Reset of the variables*/
            startExperimentation = false;
            calibrationRequest = false;
            calibrationDone = false;
            calibrationPerformed = false;
            calibrationError = false;
            leaveSequence = false;
            timeActualForCalibration = 0;

            sensor.releaseUSB();
            fileCSVIsOK = false;
            sensorIsOK = false;

            /*Turn off the state led*/
            myMotor.turnOffStateLed();

      }
```

```
		/*No command request*/
		cout << "Releasing command" << endl;

		while (true) {
			if (!myMotor.startEnable() && !myMotor.resetEnable()) {
				break;
			}
		}

		/*Start request*/
		cout << "Press <<Start>>" << endl;
		sleep(periodForMessage);

		myMotor.turnOnStartLed();

		while (true) {

			/*Blinking the led*/
			myMotor.turnOnStateLed(periodOfBlinkingLed / 10, ON);

			if (myMotor.startEnable()) {
				break;

			}
		}

		/*Turn off the state led*/
		myMotor.turnOffStateLed();

	}

	return 0;

}
```

```cpp
/*
 * Motor.cpp
 *
 *  Created on: 11 June 2015
 *      Author: Vincent Giachino
 */

#include "Motor.h"
#include <iostream>
#include <wiringPi.h>

using namespace std;

Motor::Motor() {

	/***Variables declaration***/

	/*Time and period variables*/
	periodOfCommunication = 1000; //Period of communication (in ms)

	/*Control variables*/
	pulseValue = 0; //Show the absolute value of pulses since the start

	/*Pin number declaration*/
	enableMove = 0;
	clockwiseMove = 1;
	pulseForMove = 2;
	ledState = 3;
	ledStart = 4;

	startProgram = 27;
	resetProgram = 28;
	motorReady = 29;
	positionMaxIn = 21;
	positionMaxOut = 22;
	positionZero = 23;

	/*RPI configurations*/
	wiringPiSetup();

	pinMode(enableMove, OUTPUT); //Pin 11 RPI -> Output
	pinMode(clockwiseMove, OUTPUT); //Pin 12 RPI -> Output
	pinMode(pulseForMove, OUTPUT); //Pin 13 RPI -> Output
	pinMode(ledState, OUTPUT); //Pin 15 RPI -> Output
	pinMode(ledStart, OUTPUT); //Pin 16 RPI -> Output

	pinMode(startProgram, INPUT); //Pin 36 RPI -> Input
	pinMode(resetProgram, INPUT); //Pin 38 RPI -> Input
	pinMode(motorReady, INPUT); //Pin 40 RPI -> Input
	pinMode(positionMaxIn, INPUT); //Pin 29 RPI -> Input
	pinMode(positionMaxOut, INPUT); //Pin 31 RPI -> Input
	pinMode(positionZero, INPUT); //Pin 33 RPI -> Input

	digitalWrite(enableMove, LOW); //Set the output value "LOW" to the pin 11
	digitalWrite(clockwiseMove, LOW); //Set the output value "LOW" to the pin 12
	digitalWrite(pulseForMove, LOW); //Set the output value "LOW" to the pin 13
	digitalWrite(ledState, LOW); //Set the output value "LOW" to the pin 15
	digitalWrite(ledStart, LOW); //Set the output value "LOW" to the pin 16

	pullUpDnControl(startProgram, PUD_DOWN); //Set a pull-down to the pin 36
	pullUpDnControl(resetProgram, PUD_DOWN); //Set a pull-down to the pin 38
	pullUpDnControl(motorReady, PUD_DOWN); //Set a pull-down to the pin 40
```

```
	pullUpDnControl(positionMaxIn, PUD_DOWN); //Set a pull-down to the pin 29
	pullUpDnControl(positionMaxOut, PUD_DOWN); //Set a pull-down to the pin 31
	pullUpDnControl(positionZero, PUD_DOWN); //Set a pull-down to the pin 33

	/*Driver configurations*/
	periodOfPulse = 500; //Period of pulse (in us)
	divisionPulse = 10; //Division number for one step

	/*Motor configurations*/
	pulsePerRound = 200; //Pulse per round
	advancePerRound = 2; //Advance per round (in mm)
	advancePerPulse = advancePerRound / (pulsePerRound*divisionPulse); //Advance per pulse (in mm)

}

Motor::~Motor() {
}

bool Motor::startEnable() {
	return digitalRead(startProgram);
}

bool Motor::resetEnable() {
	return digitalRead(resetProgram);
}

bool Motor::turnOnStateLed(int period, bool blinking) {

	digitalWrite(ledState, HIGH);

	if (blinking) {

		delayMicroseconds((1000 * period) / 2);
		digitalWrite(ledState, LOW);
		delayMicroseconds((1000 * period) / 2);

	}

	return true;

}

bool Motor::turnOffStateLed() {

	digitalWrite(ledState, LOW);
	return 1;

}

bool Motor::turnOnStartLed() {

	digitalWrite(ledStart, HIGH);
	return 1;

}

bool Motor::turnOffStartLed() {

	digitalWrite(ledStart, LOW);
	return 1;

}
```

```
bool Motor::movePositionZero() {

	/*Temporal variables*/
	bool tempMAZ = true; //Check if the position zero switch is already activated
	int tempNumberOfError = 0; //Count the number of times that is an error occurred

	/*Driver ignition*/
	digitalWrite(enableMove, HIGH);
	digitalWrite(clockwiseMove, BACK);

	/*Limit switch IN control*/
	if (!digitalRead(positionMaxIn)) {
		digitalWrite(clockwiseMove, FORWARD);
	}

	/*Position zero switch control*/
	if (!digitalRead(positionZero)) {
		tempMAZ = false;
	}

	/***Positioning sequences***/
	while (true) {

		/*Position zero switch control*/
		if (digitalRead(positionZero)) {
			tempMAZ = true;
		}

		/*Motor state control*/
		if (digitalRead(motorReady)) {

			/*Motion control*/
			digitalWrite(pulseForMove, HIGH);
			delayMicroseconds(periodOfPulse);
			digitalWrite(pulseForMove, LOW);
			delayMicroseconds(periodOfPulse);

		}

		/*Position zero is done*/
		if (!digitalRead(positionZero) && digitalRead(clockwiseMove)
				&& tempMAZ) {

			digitalWrite(enableMove, LOW);
			return true;

		}

		/*Position zero is done, but since the wrong side*/
		if (!digitalRead(positionZero) && !digitalRead(clockwiseMove)) {

			while (!digitalRead(positionZero)) {
				move(5, FORWARD, OFF);
			}

			move(5, FORWARD, OFF);

			digitalWrite(clockwiseMove, BACK);
			digitalWrite(enableMove, HIGH);
			delay(periodOfCommunication);
```

```
            }

            /*Limit switch IN control*/
            if (!digitalRead(positionMaxIn)) {

                  digitalWrite(clockwiseMove, FORWARD);
                  delay(periodOfCommunication);

            }

            /*Limit switch OUT control*/
            if (!digitalRead(positionMaxOut)) {

                  digitalWrite(clockwiseMove, BACK);
                  delay(periodOfCommunication);

            }

            /*Error control*/
            if (!digitalRead(motorReady)) {

                  digitalWrite(enableMove, LOW);
                  cout << "Error : Motor is not ready" << endl;
                  delay(periodOfCommunication);
                  digitalWrite(enableMove, HIGH);

                  tempNumberOfError++;

                  if (tempNumberOfError >= 500) {

                        cout << "Error : Move is stopped" << endl;
                        delay(periodOfCommunication);

                        digitalWrite(enableMove, LOW);
                        return false;

                  }

            }

            /*Reset program control*/
            if (digitalRead(resetProgram)) {

                  cout << "Error : Program is reseted" << endl;
                  delay(periodOfCommunication);

                  digitalWrite(enableMove, LOW);
                  return false;

            }

      }

      /*Liberation driver*/
      digitalWrite(enableMove, LOW);
      return true;

}

bool Motor::move(double distance, bool side, bool calibration) {

      /*Temporal variables*/
```

```
	bool tempMaxInOut = false; //Check if the limit switches IN or OUT are activated
	int tempNumberOfError = 0; //Count the number of times that is an error occurred
	int tempNumberOfPulse = distance / advancePerPulse; //Calculated the numbers of pulse to join the position

	/*Driver ignition*/
	digitalWrite(enableMove, HIGH);
	digitalWrite(clockwiseMove, side);

	/***Positioning sequences***/
	for (int i = tempNumberOfPulse; i > 0;) {

		/*Limit switch IN control*/
		if (!digitalRead(positionMaxIn) && digitalRead(clockwiseMove)) {

			tempMaxInOut = true;

			i = 0;

			cout << "Position is maximum IN" << endl;
			delay(periodOfCommunication);

			digitalWrite(enableMove, LOW);
			return false;

		}

		/*Limit switch OUT control*/
		if (!digitalRead(positionMaxOut) && !digitalRead(clockwiseMove)) {

			tempMaxInOut = true;

			i = 0;

			cout << "Position is maximum OUT" << endl;
			delay(periodOfCommunication);

			digitalWrite(enableMove, LOW);
			return false;

		}

		/*Motor state control*/
		if (digitalRead(motorReady) && !tempMaxInOut) {

			/*Motion control*/
			digitalWrite(pulseForMove, HIGH);
			delayMicroseconds(periodOfPulse);
			digitalWrite(pulseForMove, LOW);
			delayMicroseconds(periodOfPulse);

			i--;

		}

		/*Error control*/
		if (!digitalRead(motorReady)) {

			digitalWrite(enableMove, LOW);
			cout << "Error : Motor is not ready" << endl;
			delay(periodOfCommunication);
			digitalWrite(enableMove, HIGH);
```

```
                tempNumberOfError++;

                if (tempNumberOfError >= 500) {

                        cout << "Error : Move is stopped" << endl;
                        delay(periodOfCommunication);

                        i = 0;

                        digitalWrite(enableMove, LOW);
                        return false;

                }

        }

        /*Reset program control*/
        if (digitalRead(resetProgram) && !calibration) {

                delay(periodOfCommunication);
                cout << "Error : Program is reseted" << endl;

                digitalWrite(enableMove, LOW);
                return false;

        }

    }

    /*Liberation driver*/
    digitalWrite(enableMove, LOW);
    return true;

}

bool Motor::move(int pulse, bool side) {

    /*Temporal variables*/
    bool tempMaxInOut = false; //Check if the limit switches IN or OUT are activated
    int tempNumberOfError = 0; //Count the number of times that is an error occurred

    /*Driver ignition*/
    digitalWrite(enableMove, HIGH);
    digitalWrite(clockwiseMove, side);

    /***Positioning sequences***/
    for (int i = pulse; i > 0;) {

        /*Limit switch IN control*/
        if (!digitalRead(positionMaxIn) && digitalRead(clockwiseMove)) {

                tempMaxInOut = true;

                i = 0;

                cout << "Position is maximum IN" << endl;
                delay(periodOfCommunication);

                digitalWrite(enableMove, LOW);
                return false;
```

```
}

/*Limit switch OUT control*/
if (!digitalRead(positionMaxOut) && !digitalRead(clockwiseMove)) {

	tempMaxInOut = true;

	i = 0;

	cout << "Position is maximum OUT" << endl;
	delay(periodOfCommunication);

	digitalWrite(enableMove, LOW);
	return false;

}

/*Motor state control*/
if (digitalRead(motorReady) && !tempMaxInOut) {

	/*Motion control*/
	digitalWrite(pulseForMove, HIGH);
	delayMicroseconds(periodOfPulse);
	digitalWrite(pulseForMove, LOW);
	delayMicroseconds(periodOfPulse);

	i--;

}

/*Error control*/
if (!digitalRead(motorReady)) {

	digitalWrite(enableMove, LOW);
	cout << "Error : Motor is not ready" << endl;
	delay(periodOfCommunication);
	digitalWrite(enableMove, HIGH);

	tempNumberOfError++;

	if (tempNumberOfError >= 500) {

		cout << "Error : Move is stopped" << endl;
		delay(periodOfCommunication);

		i = 0;

		digitalWrite(enableMove, LOW);
		return false;

	}

}

/*Reset program control*/
if (digitalRead(resetProgram)) {

	delay(periodOfCommunication);
	cout << "Error : Program is reseted" << endl;

	digitalWrite(enableMove, LOW);
	return false;
```

```
            }

        }

        /*Liberation driver*/
        digitalWrite(enableMove, LOW);
        return true;

}
```

```
/*
 * Motor.h
 *
 *  Created on: 11 June 2015
 *      Author: Vincent Giachino
 */

#ifndef MOTOR_H_
#define MOTOR_H_

#define FORWARD          0
#define BACK             1
#define ON               1
#define OFF              0

class Motor {

public:

	/***Variables declaration***/

	/*Time and period variables*/
	int periodOfCommunication; //Period of communication (in ms)

	/*Control variables*/
	int pulseValue; //Show the absolute value of pulses since the start

	/*Pin number declaration*/
	int enableMove;
	int clockwiseMove;
	int pulseForMove;
	int ledState;
	int ledStart;

	int startProgram;
	int resetProgram;
	int positionMaxIn;
	int positionMaxOut;
	int positionZero;
	int motorReady;

	/*Driver configurations*/
	int periodOfPulse; //Period of pulse (in us)
	int divisionPulse; //Division number for one step

	/*Motor configurations*/
	int pulsePerRound; //Pulse per round
	double advancePerRound; //Advance per round (in mm)
	double advancePerPulse; //Advance per pulse (in mm)

public:

	Motor();
	virtual ~Motor();

	bool startEnable(); // Return the value of the start button
	bool resetEnable(); // Return the value of the reset button
	bool turnOnStateLed(int period, bool blinking); // Turn on the state led (blinking : 0 -> no
blinking - blinking : 1 -> blinking)
	bool turnOffStateLed(); // Turn off the state led
	bool turnOnStartLed(); // Turn on the start led
	bool turnOffStartLed(); // Turn off the start led
```

```
      bool movePositionZero(); // Move the sensor to the zero position
      bool move(double distance, bool side, bool calibration); // Move the sensor to selected
distance (side : 0 -> forward - side : 1 -> back)
      bool move(int pulse, bool side); // Move the sensor to selected number of pulse (side : 0 ->
forward - side : 1 -> back)

};

#endif /* MOTOR_H_ */
```

```
/*
 * USBTransfer.cpp
 *
 *  Created on: 15 June 2015
 *      Author: Vincent Giachino
 */

#include "USBTransfer.h"
#include <iostream>
#include <math.h>
#include <string.h>
#include <libusb-1.0/libusb.h>

using namespace std;

/*Initialize the devices and the context*/
libusb_device **device; //Pointer to pointer of device, used to retrieve a list of devices
libusb_device_handle *deviceHandle; //Get a device handle
libusb_context *context = NULL; //Get a libUSB session

/*Variables declaration*/
int value;
ssize_t count;

USBTransfer::USBTransfer() {

	/*USB connection declaration*/
	vidNumber = 0x0661; //Vendor identification
	pidNumber = 0x3706; //Product identification

}

USBTransfer::~USBTransfer() {
}

int USBTransfer::initUSB() {

	/*Initialize the library*/
	value = libusb_init(&context); //Initialize the library for the session we have just declared

	/*Check if there is an error*/
	if (value < 0) {

		cout << "Initialization error " << value << endl;
		return 0;

	}

	/*Configuration mode*/
	libusb_set_debug(context, 3); //Set verbosity level to 3 (as suggested in the documentation)

	/*Get the devices list*/
	count = libusb_get_device_list(context, &device);

	/*Check if there is an error*/
	if (count < 0) {

		cout << "Get device error" << endl;
		return 0;

	}
```

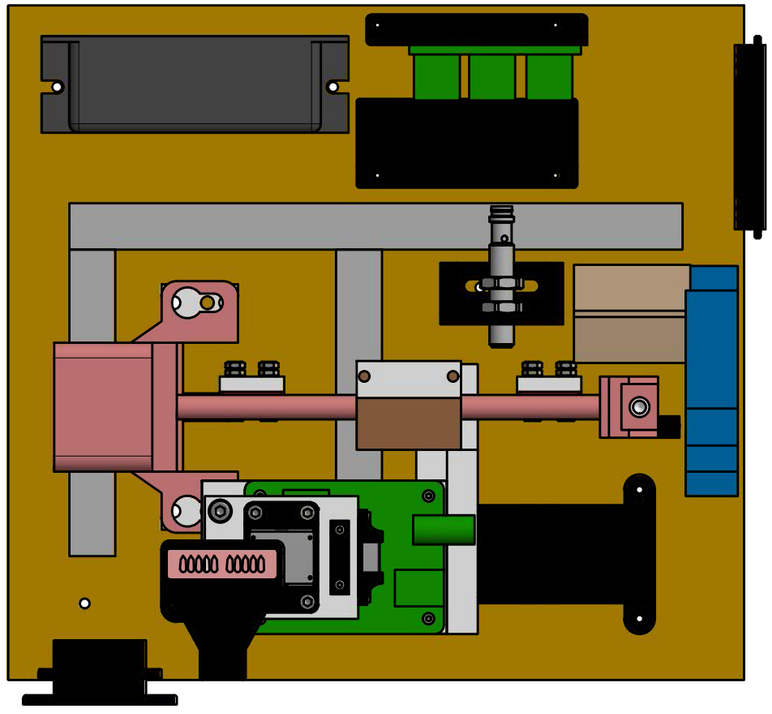
```
	/*Display the number of devices*/
	cout << count << " Devices in list" << endl;

	/*Open the devices*/
	deviceHandle = libusb_open_device_with_vid_pid(context, vidNumber,
			pidNumber);

	/*Check if there is an error*/
	if (deviceHandle == NULL) {

		cout << "Can not open device" << endl;
		return 0;

	} else {
		cout << "Device opened" << endl;
	}

	/*List free*/
	libusb_free_device_list(device, 1);
	return 1;

}

int USBTransfer::sendData(int fonction) {

	/*Variables declaration*/
	int length;
	int actual;
	unsigned char *data;

	/*Select the data to send*/
	switch (fonction) {

	case setDefinedHV: //Define the high voltage value (1000V)

		data = new unsigned char[2];
		data[0] = 0x44;
		data[1] = 0x56;

		length = 2;

		break;

	case setDefinedLLD: //Define the low voltage value (-V)

		data = new unsigned char[2];
		data[0] = 0x44;
		data[1] = 0x4C;

		length = 2;

		break;
	case setIntTime: //Define the integration time (10s)

		data = new unsigned char[8];
		data[0] = 0x49;
		data[1] = 0x00;
		data[2] = 0x00;
		data[3] = 0x00;
		data[4] = 0x40;
		data[5] = 0x42;
		data[6] = 0x0F;
```

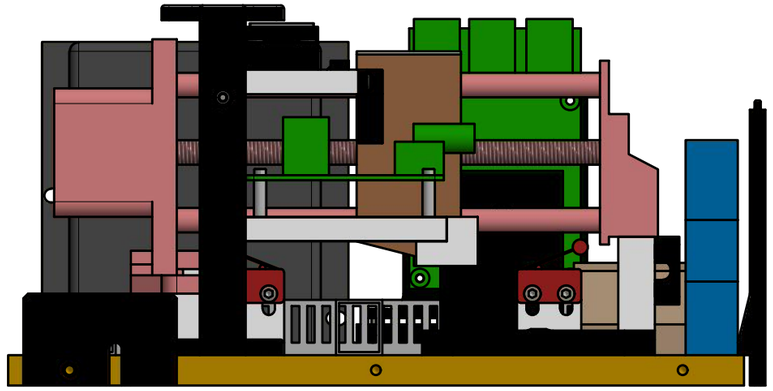
```
		data[7] = 0x00;

		length = 8;

		break;
	case setRepeat: //Set the repeat mode

		data = new unsigned char[8];
		data[0] = 0x52;
		data[1] = 0x00;
		data[2] = 0x00;
		data[3] = 0x00;
		data[4] = 0x00;
		data[5] = 0x00;
		data[6] = 0x00;
		data[7] = 0x00;

		length = 8;

		break;
	case startCount: //Start the counting

		data = new unsigned char[1];
		data[0] = 0x43;

		length = 1;

		break;

	case stopCount: //Stop the counting

		data = new unsigned char[1];
		data[0] = 0x0d;

		length = 1;

		break;

	case cutHV: //Cut the high voltage

		data = new unsigned char[8];
		data[0] = 0x56;
		data[1] = 0x00;
		data[2] = 0x00;
		data[3] = 0x00;
		data[4] = 0x00;
		data[5] = 0x00;
		data[6] = 0x00;
		data[7] = 0x00;

		length = 8;

		break;

	default:
		break;

	}

	/*Kernel configuration*/
	if (libusb_kernel_driver_active(deviceHandle, 0) == 1) { //Find out if kernel driver is
attached
```

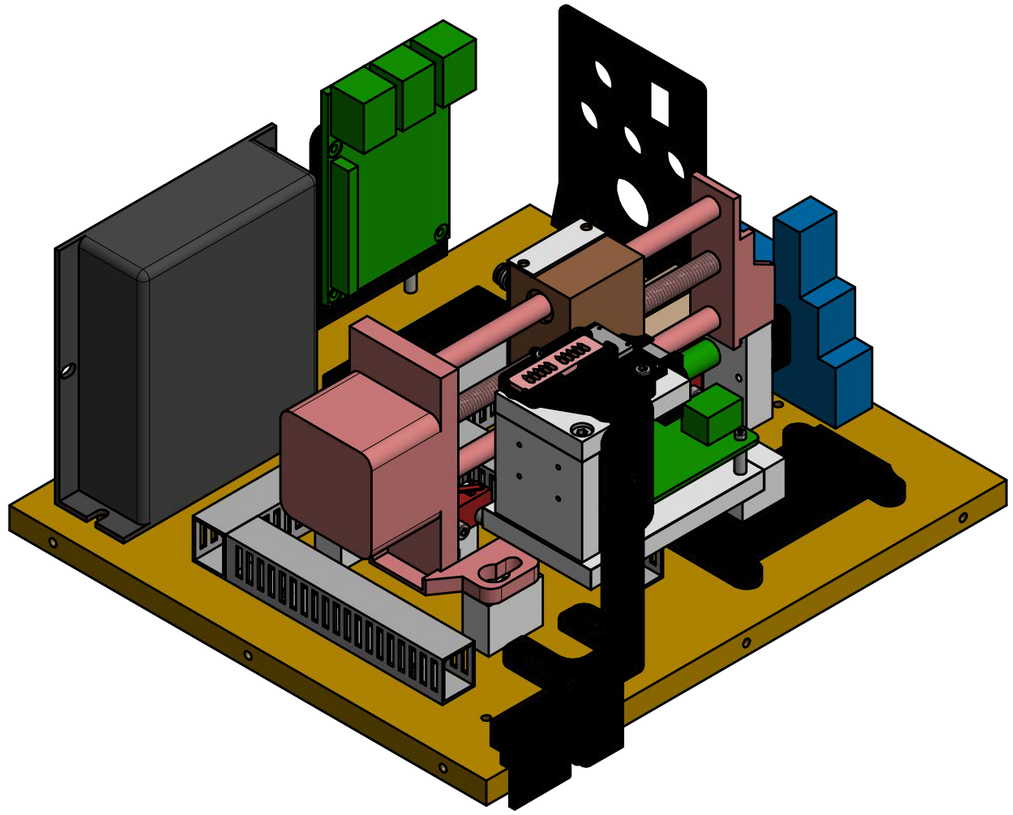
```
		if (libusb_detach_kernel_driver(deviceHandle, 0) != 0) {
			cout << "Kernel driver not detached" << endl; //Detach the kernel driver
		}

	}

	/*Claim configuration*/
	value = libusb_claim_interface(deviceHandle, 0); //Claim interface 0 (the first) of device
(mine had just 1)

	/*Check if there is an error*/
	if (value < 0) {

		cout << "Cannot claim interface" << endl;

		delete[] data; //Delete the allocated memory for data
		return 0;

	}

	/*Transfer the data*/
	value = libusb_bulk_transfer(deviceHandle, (1 | LIBUSB_ENDPOINT_OUT), data,
			length, &actual, 0);

	/*Check if the message is successfully sent*/
	if (!(value == 0 && actual == length)) {

		cout << "Writing error" << endl;
		delete[] data; //Delete the allocated memory for data
		return 0;

	}

	/*Delete the allocated memory for data*/
	delete[] data;
	return 1;

}

int USBTransfer::readData(unsigned char data[], int dataLength) {

	/*Variables declaration*/
	int actual = 0;

	/*Receive the data*/
	libusb_bulk_transfer(deviceHandle, (1 | LIBUSB_ENDPOINT_IN),
			(unsigned char*) data, dataLength, &actual, 0);

	return 1;

}

double USBTransfer::sequenceOfTransfer(USBTransfer sensor, int headerLength,
		int dataLength) {

	unsigned char data[dataLength];
	unsigned char dataTemp[dataLength - headerLength];

	double tempValue = 0;

	sensor.sendData(setDefinedHV);
```

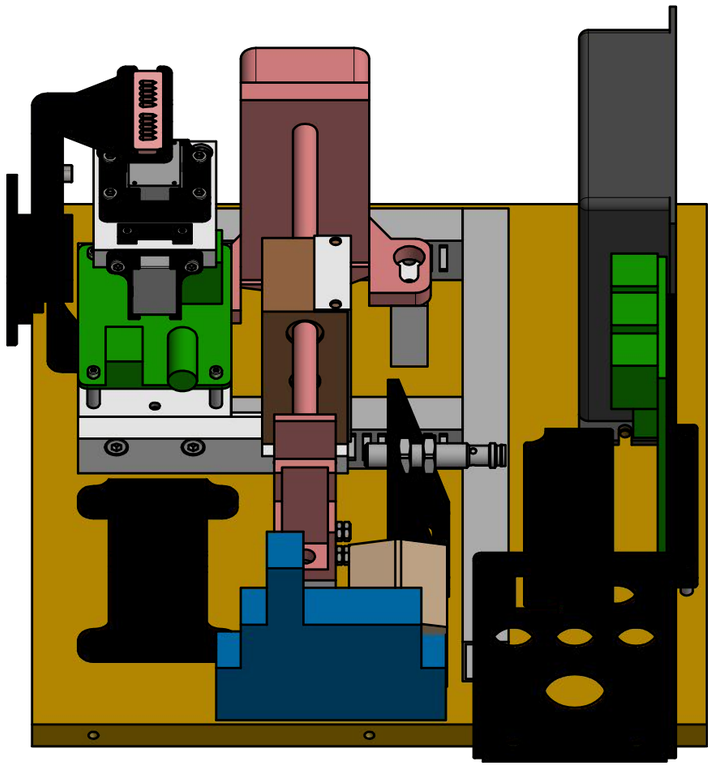
```
	sensor.readData(data, dataLength);

	sensor.sendData(setDefinedLLD);
	sensor.readData(data, dataLength);

	sensor.sendData(setIntTime);
	sensor.readData(data, dataLength);

	sensor.sendData(setRepeat);
	sensor.readData(data, dataLength);

	sensor.sendData(startCount);
	sensor.readData(data, dataLength);

	memcpy(dataTemp, data + headerLength, dataLength - headerLength);

	sensor.sendData(stopCount);

	sensor.sendData(cutHV);
	sensor.readData(data, dataLength);

	for (int i = 0; i < (dataLength - headerLength); i++) {

		if (dataTemp[i] > 0) {
			tempValue += (double) (dataTemp[i] * pow(256.0, i));
		}

	}

	return tempValue;

}

int USBTransfer::releaseUSB() {

	/*Release the claimed interface*/
	value = libusb_release_interface(deviceHandle, 0);

	/*Check if there is an error*/
	if (value != 0) {

		cout << "Can not release interface" << endl;
		return 0;

	}

	cout << "Released interface" << endl;

	/*Close the device that is opened*/
	libusb_close(deviceHandle);
	libusb_exit(context);

	return 1;

}
```

```
/*
 * USBTransfer.h
 *
 *  Created on: 15 June 2015
 *      Author: Vincent Giachino
 */

#ifndef USBTRANSFER_H_
#define USBTRANSFER_H_

/*Variables declaration*/
#define      setDefinedHV             0
#define      setDefinedLLD            1
#define      setIntTime               2
#define      setRepeat                3
#define      startCount               4
#define      stopCount                5
#define      cutHV                    6

class USBTransfer {

public:

      /*USB connection declaration*/
      int vidNumber;
      int pidNumber;

public:

      USBTransfer();
      virtual ~USBTransfer();

      int initUSB(); //Initialized the USB transfer connection
      int sendData(int fonction); //Send data to the     host
      int readData(unsigned char data[], int dataLength); //Receive data from the host
      double sequenceOfTransfer(USBTransfer sensor, int headerLength,
                  int dataLength); //Create the sequence for the measurement
      int releaseUSB(); //Released the USB transfer connection

};

#endif /* USBTRANSFER_H_ */
```

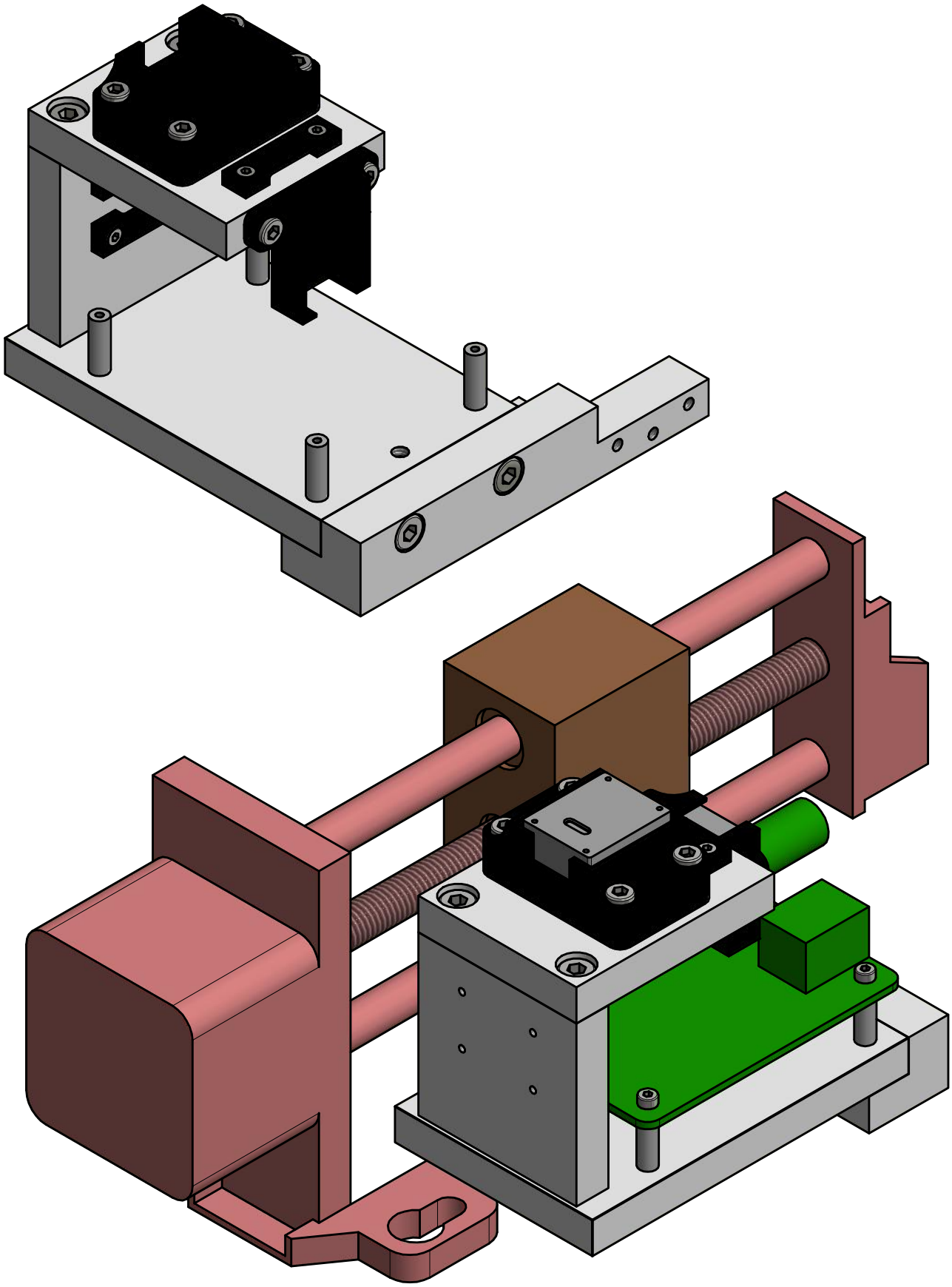






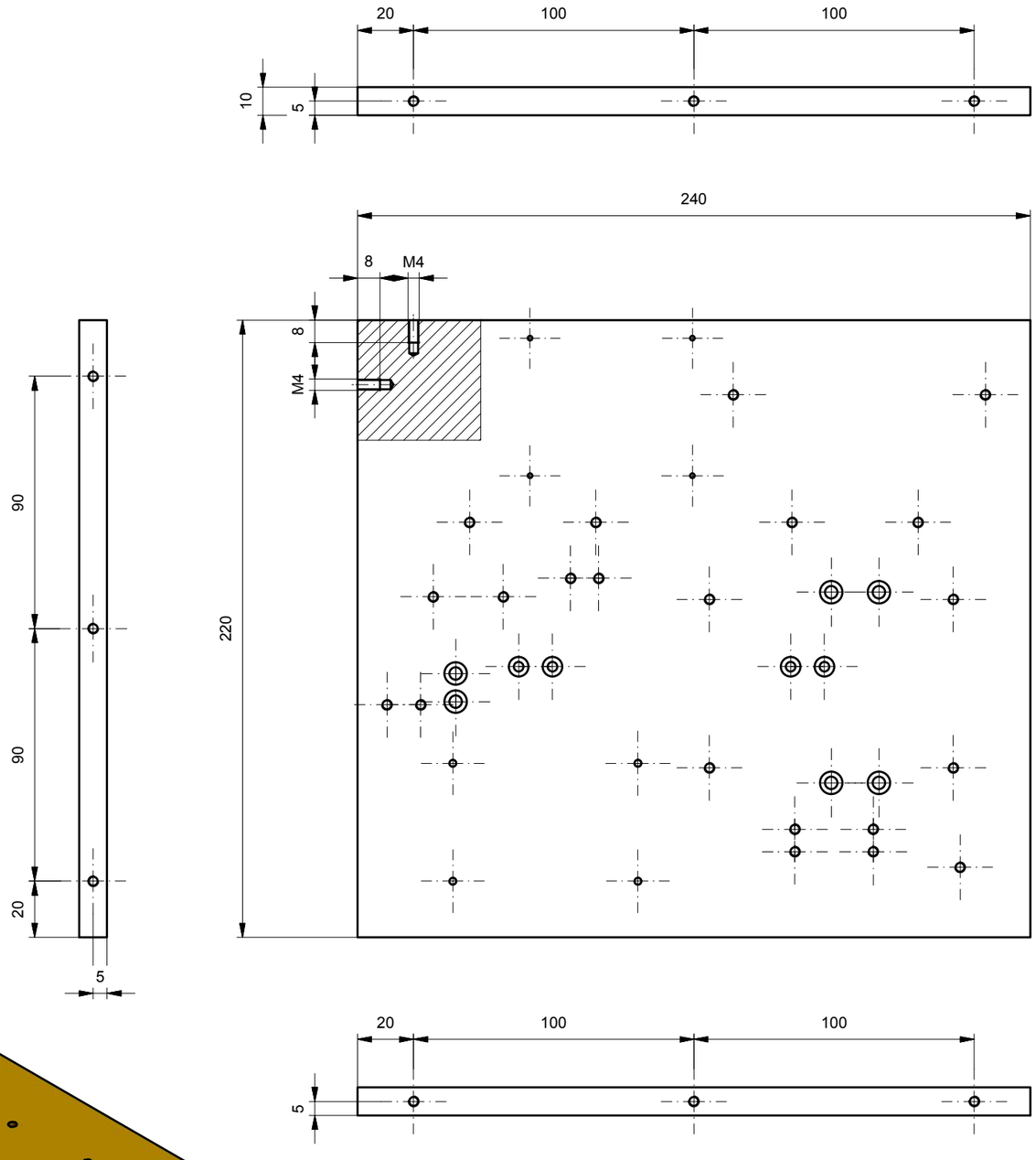


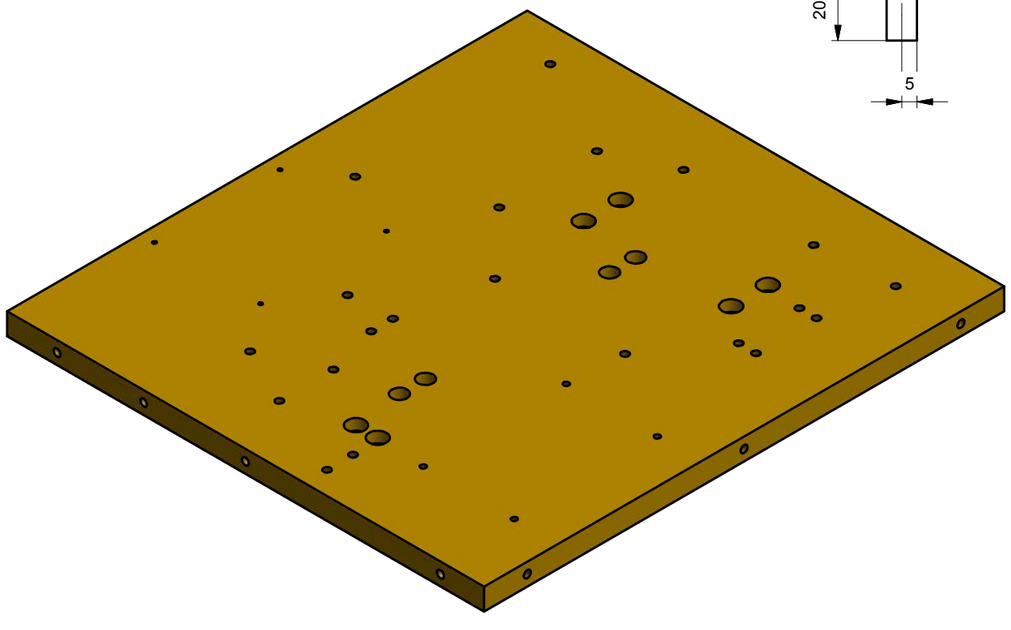
Système complet Projet BRAAVOO	Dessiné Gezeichnet	Vincent Giachino	08.06.2015	Echelle Massstab
	Contrôlé Geprüft			1:2
Fichier Datei U:\PRD\Composants\Assemblage\Systeme_Complet.idw				
Hes·so VALAIS WALLIS	1			



Système de mesure Projet BRAAVOO	Dessiné Gezeichnet	Vincent Giachino	07.07.2015	Echelle Massstab 1:1
	Contrôlé Geprüft			
Fichier Datei U:\PRD\Composants\Assemblage\Systeme_De_Mesure.idw				
Hes·so VALAIS WALLIS	1			

Perçages latéraux (M4)





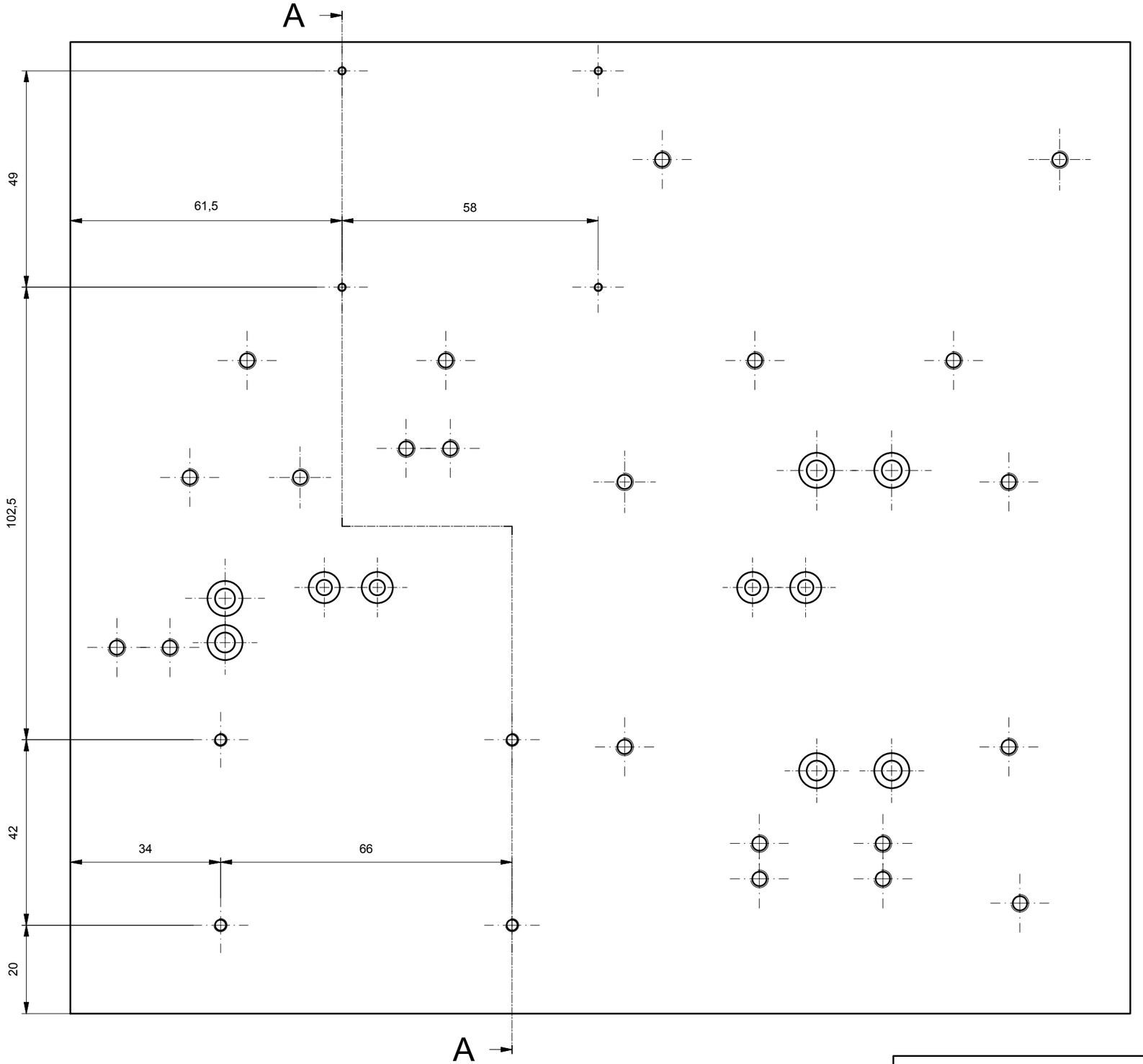
Pos. Pos.	Quantité Menge	Matière Werkstoff	Dénomination / Caractéristiques Benennung / Merkmale
P001	1	Aluminium	Chassis

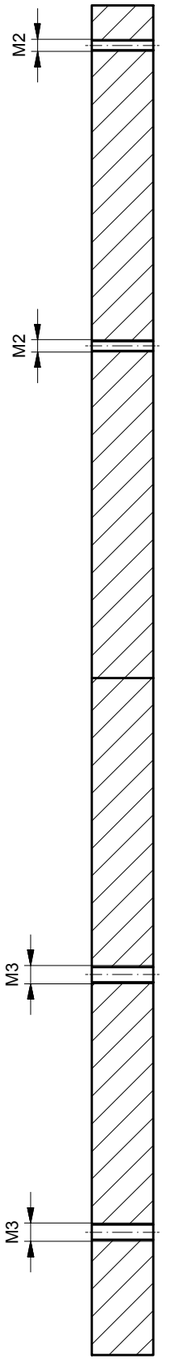
Chassis Projet BRAAVOO	Dessiné Gezeichnet	Vincent Giachino	09.06.2015	Echelle Massstab 1:2
	Contrôlé Geprüft			

Fichier
Datei U:\PRD\Composants\Carcasse\Chassis_Percages_Lateraux.idw

Hes·so VALAIS WALLIS

1

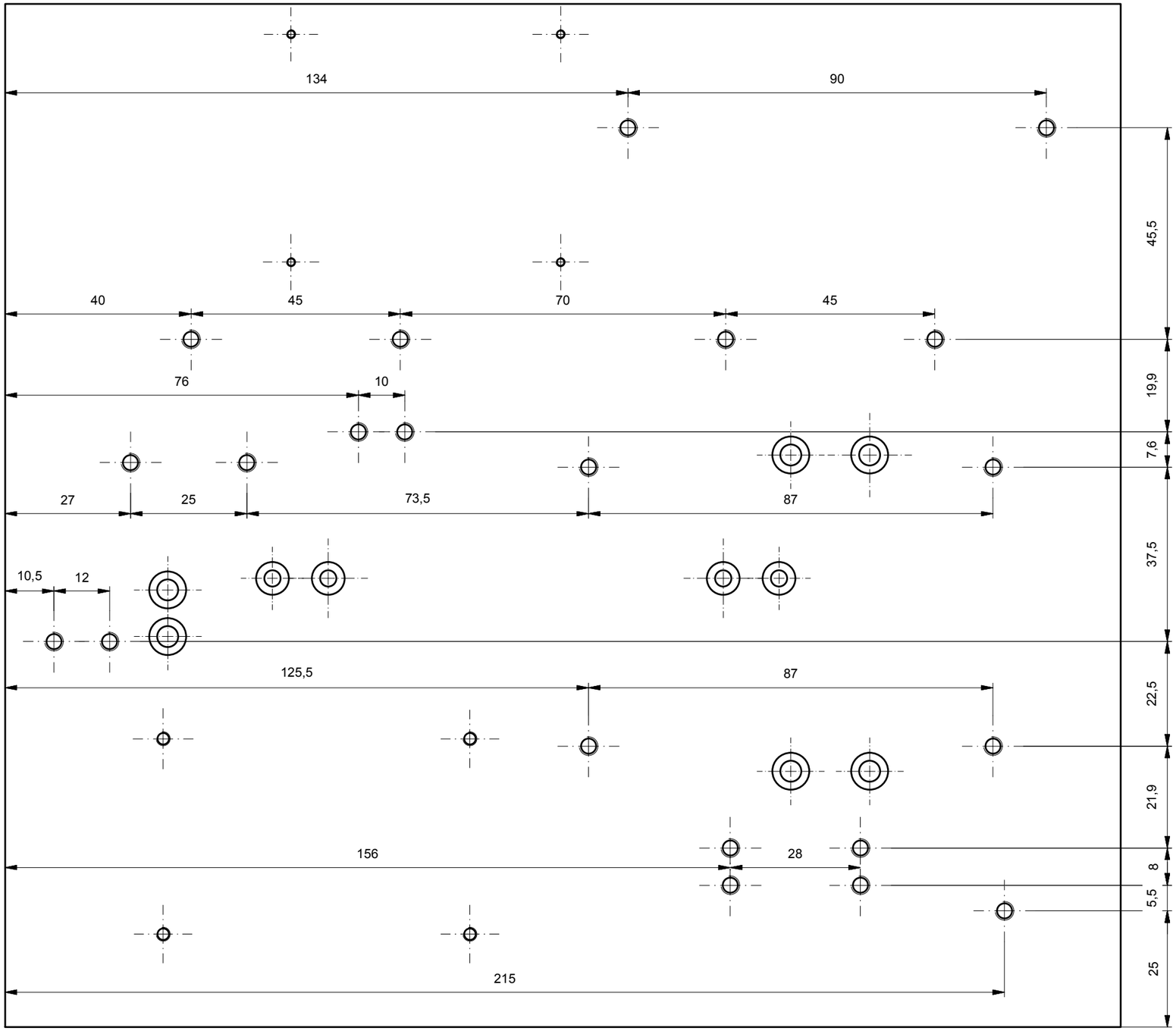




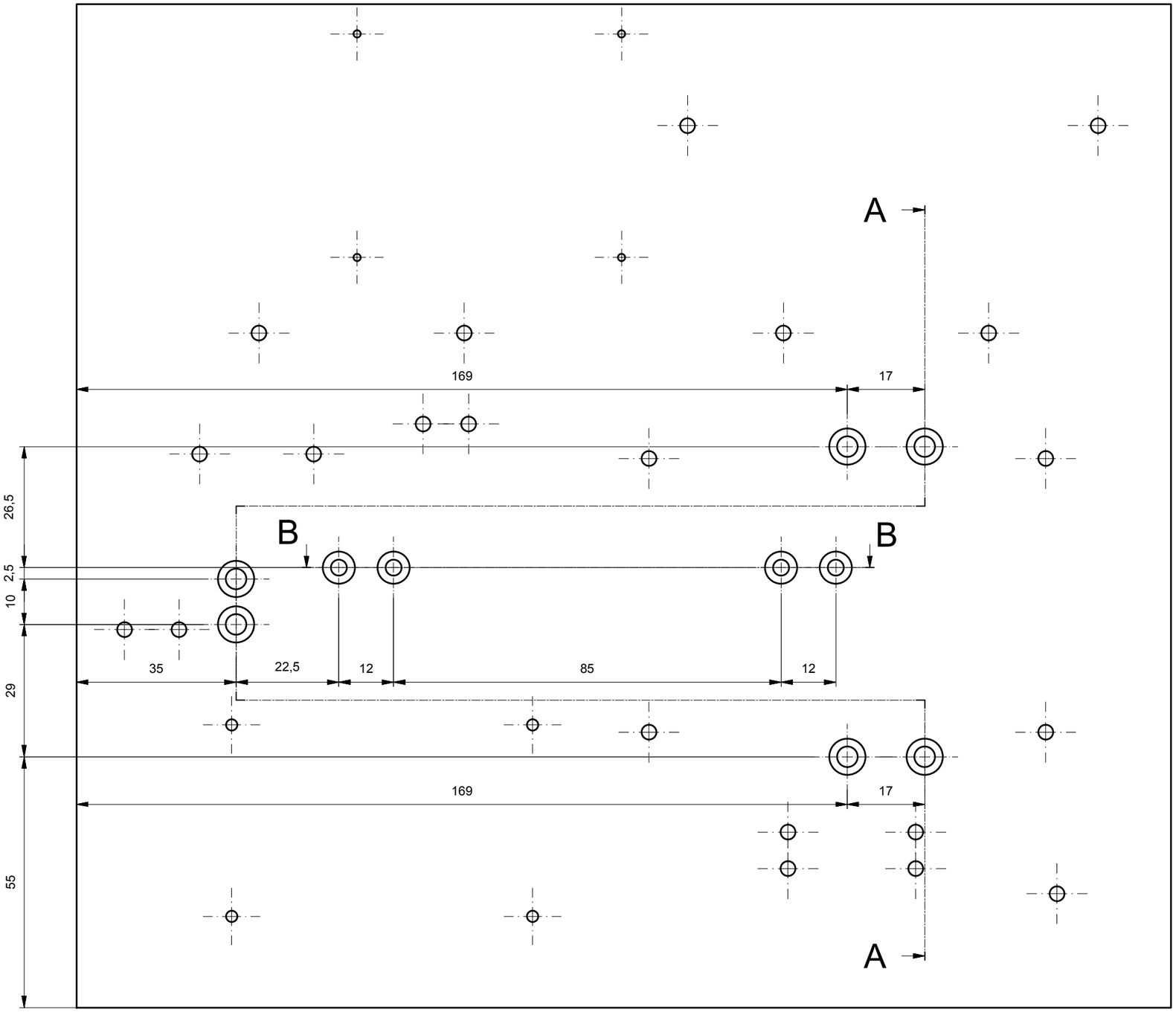
A-A (1 : 1)

Chassis Projet BRAAVOO	Dessiné Gezeichnet	Vincent Giachino	09.06.2015	Echelle Massstab 1:1
	Contrôlé Geprüft			
Fichier Datei U:\PRD\Composants\Carcasse\Chassis_Percages_M2_M3.idw				
Hes·so VALAIS WALLIS	2			

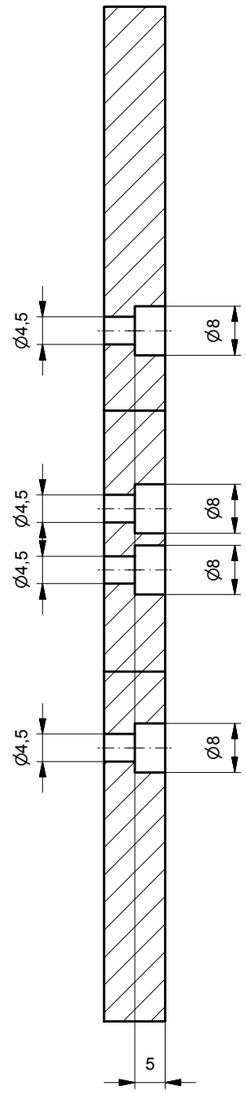
Perçages M4 (traversants)



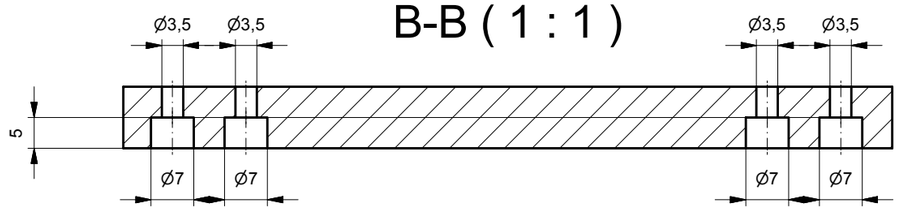
Chassis Projet BRAAVOO	Dessiné Gezeichnet	Vincent Giachino	09.06.2015	Echelle Massstab
	Contrôlé Geprüft			1:1
Fichier Datei	U:\PRD\Composants\Carcasse\Chassis_Percages_M4.idw			
Hes·so VALAIS WALLIS	3			



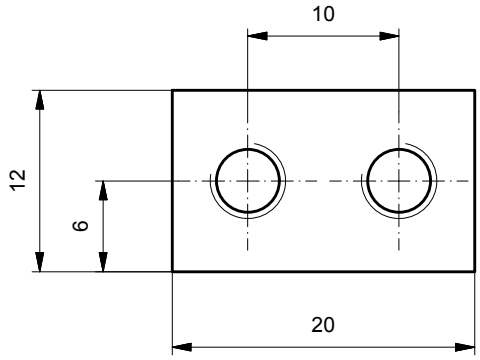
A-A (1 : 1)

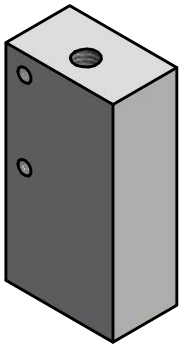


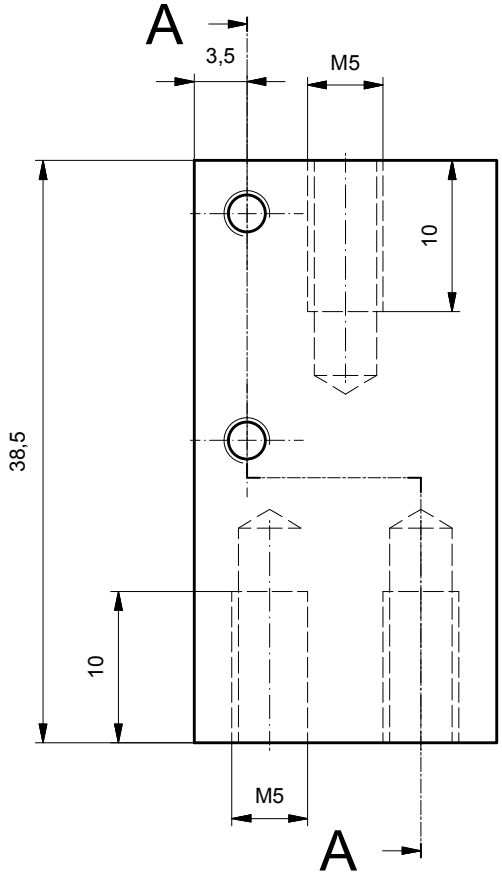
B-B (1 : 1)



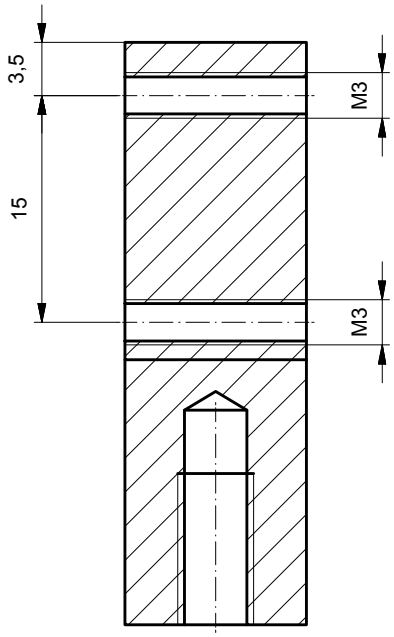
Chassis Projet BRAAVOO	Dessiné Gezeichnet	Vincent Giachino	09.06.2015	Echelle Massstab 1:1
	Contrôlé Geprüft			
Fichier Datei	U:\PRD\Composants\Carcasse\Chassis_Trous_De_Passage.idw			
Hes·so VALAIS WALLIS	4			

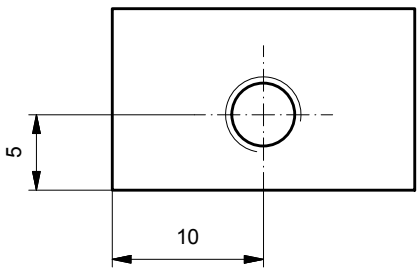






A-A (2 : 1)





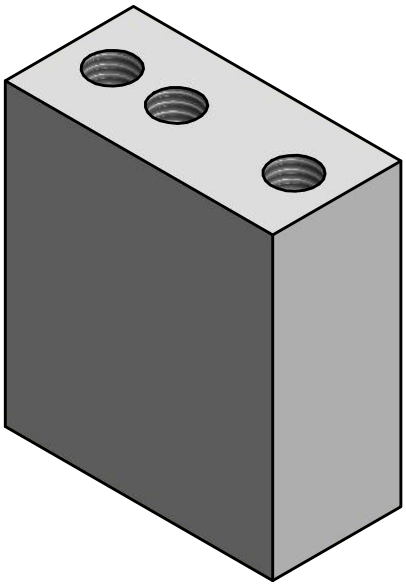
Pos. Pos.	Quantité Menge	Matière Werkstoff	Dénomination / Caractéristiques Benennung / Merkmale
P001	1	Aluminium	Plot du moteur (EST)

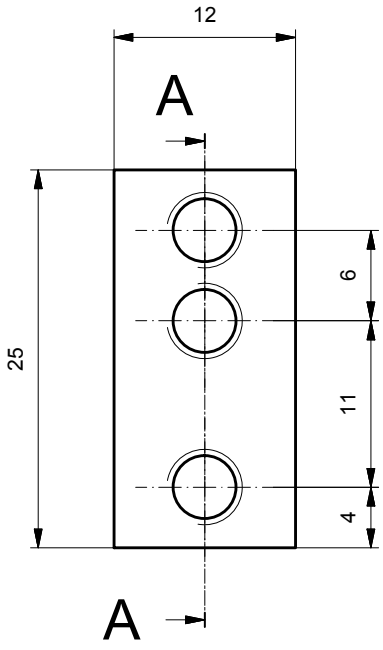
Plot du moteur (EST) Projet BRAAVOO	Dessiné Gezeichnet	Vincent Giachino	08.06.2015	Echelle Massstab 2:1
	Contrôlé Geprüft			

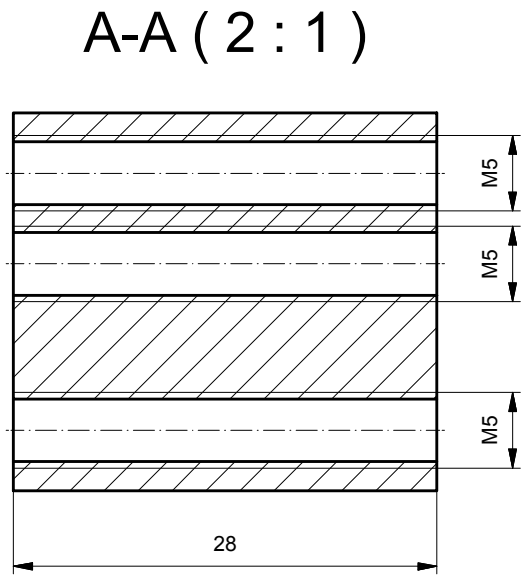
Fichier
Datei U:\PRD\Composants\Plot\Plot_Moteur_Est.idw

Hes·so VALAIS WALLIS

1







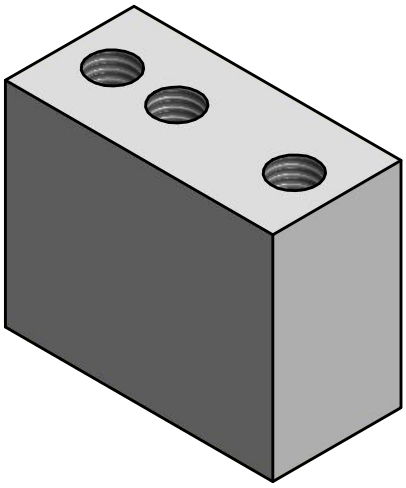
Pos. Pos.	Quantité Menge	Matière Werkstoff	Dénomination / Caractéristiques Benennung / Merkmale
P001	1	Aluminium	Plot du moteur (NORD)

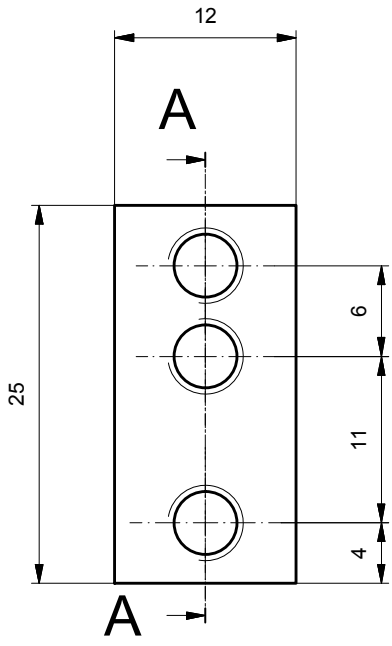
Plot du moteur (NORD) Projet BRAAVOO	Dessiné Gezeichnet	Vincent Giachino	08.06.2015	Echelle Massstab 2:1
	Contrôlé Geprüft			

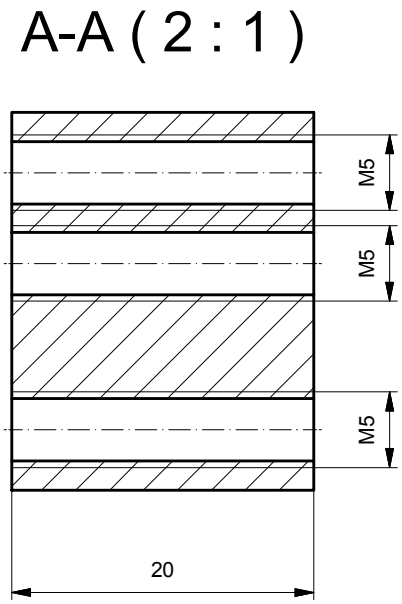
Fichier
Datei U:\PRD\Composants\Plot\Plot_Moteur_Nord.idw

Hes·so VALAIS WALLIS

1

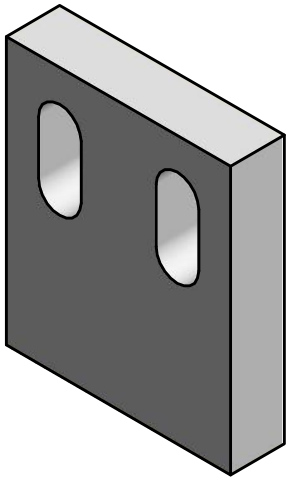


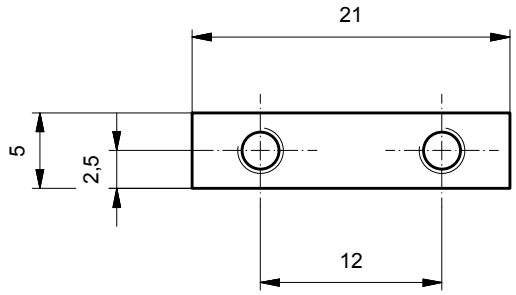




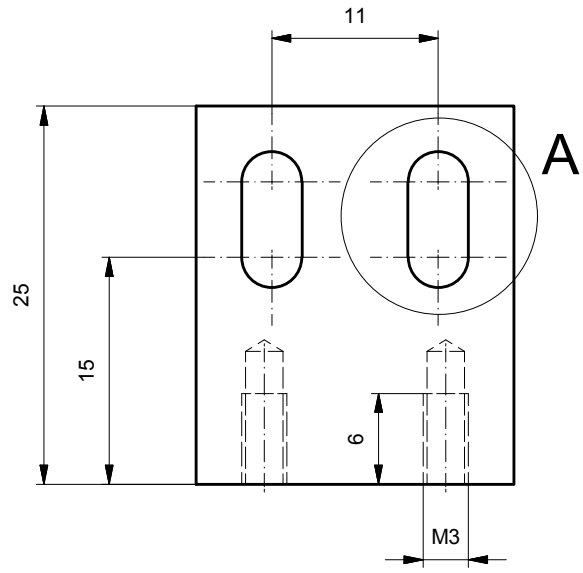
Pos. Pos.	Quantité Menge	Matière Werkstoff	Dénomination / Caractéristiques Benennung / Merkmale
P001	1	Aluminium	Plot du moteur (SUD)

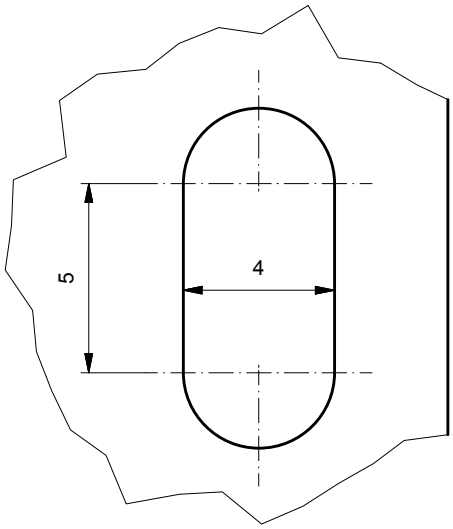
Plot du moteur (SUD) Projet BRAAVOO	Dessiné Gezeichnet	Vincent Giachino	08.06.2015	Echelle Massstab
	Contrôlé Geprüft			2:1
Fichier Datei U:\PRD\Composants\Plot\Plot_Moteur_Sud.idw				
Hes·so VALAIS WALLIS	1			





A (5 : 1)



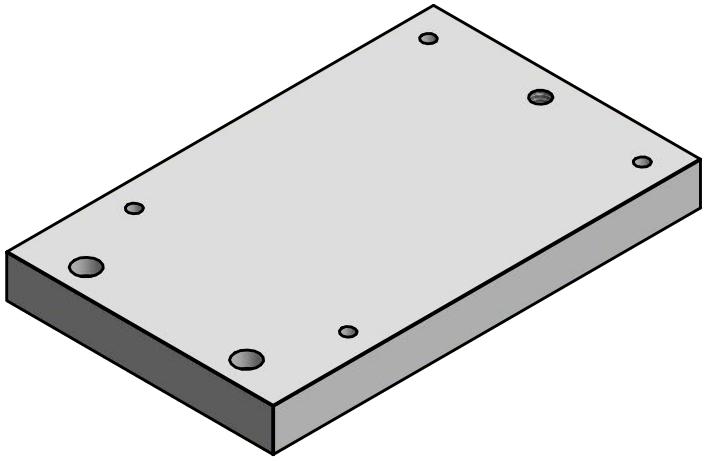


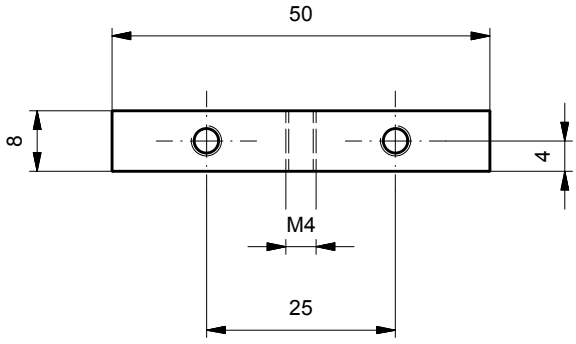
Pos. Pos.	Quantité Menge	Matière Werkstoff	Dénomination / Caractéristiques Benennung / Merkmale
P001	2	Aluminium	Support du fin de course

Support du fin de course	Dessiné Gezeichnet	Vincent Giachino	08.06.2015	Echelle Massstab
Projet BRAAVOO	Contrôlé Geprüft			2:1

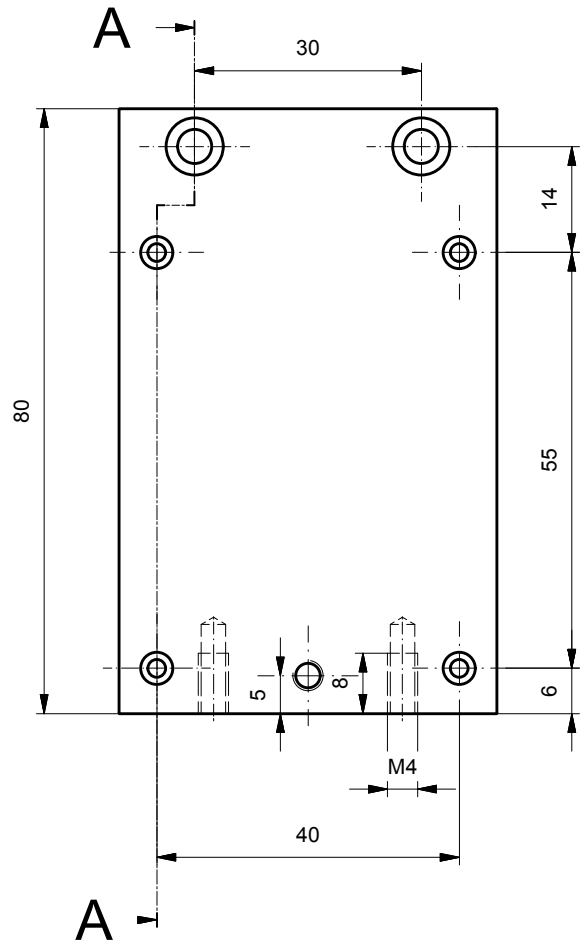
Fichier
Datei U:\PRD\Composants\Fin de course\Support_Fin_De_Course.idw

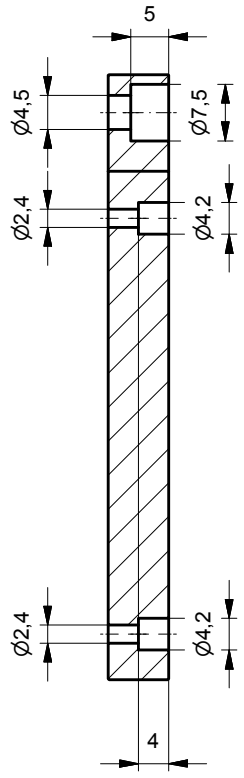
Hes·so VALAIS WALLIS





A-A (1 : 1)



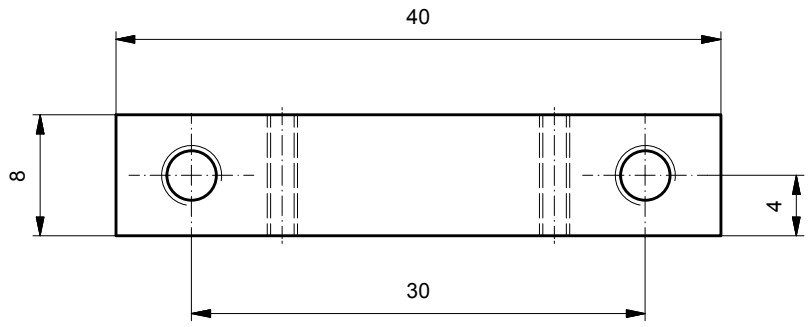


Pos. Pos.	Quantité Menge	Matière Werkstoff	Dénomination / Caractéristiques Benennung / Merkmale
P001	1	Aluminium	Socle

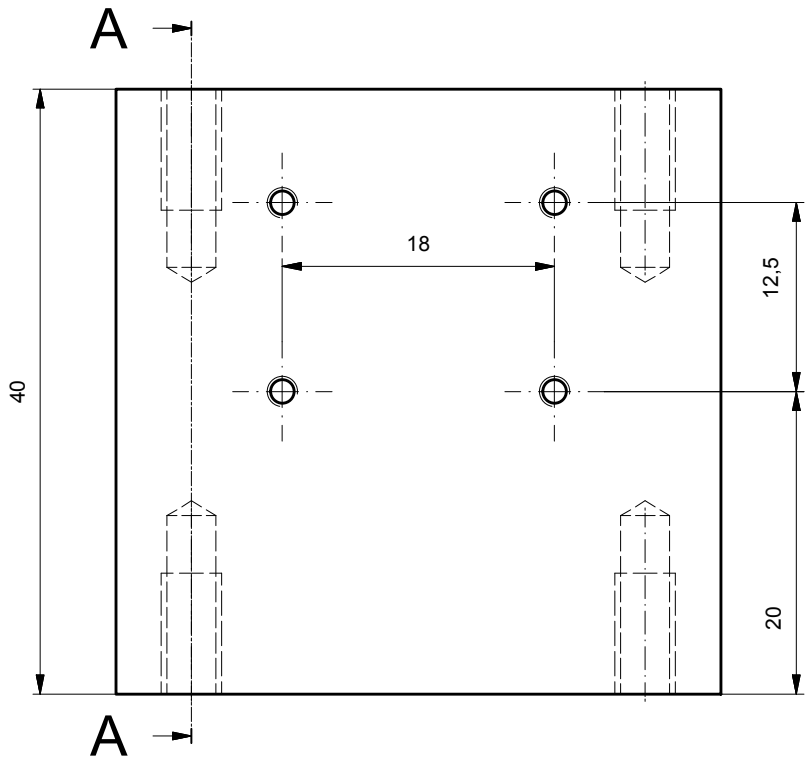
Socle Projet BRAAVOO	Dessiné Gezeichnet	Vincent Giachino	02.06.2015	Echelle Massstab
	Contrôlé Geprüft			1:1

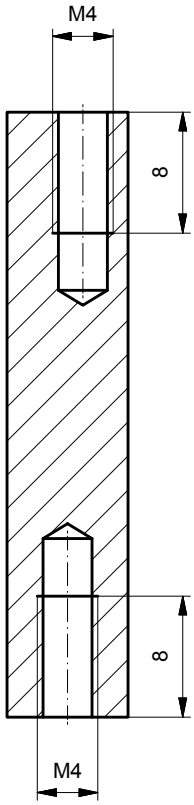
Fichier
Datei U:\PRD\Composants\Support de mesure\Socle.idw

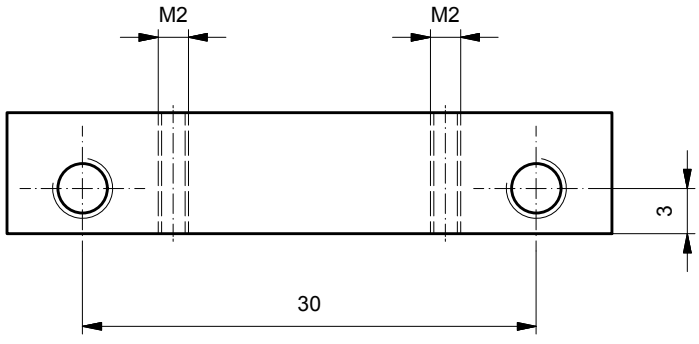
Hes·so VALAIS WALLIS

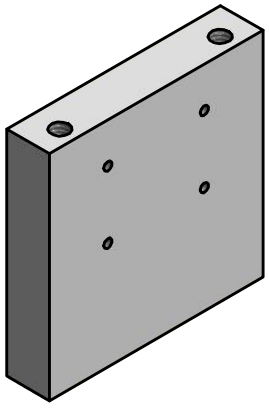


A-A (2 : 1)







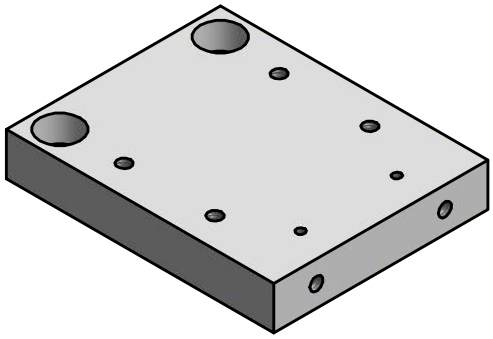


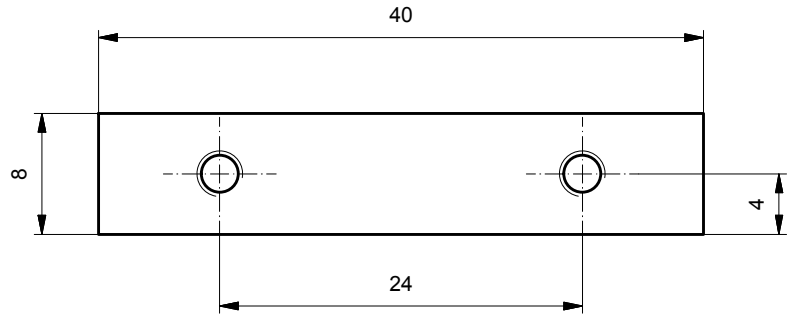
Pos. Pos.	Quantité Menge	Matière Werkstoff	Dénomination / Caractéristiques Benennung / Merkmale
P001	1	Aluminium	Support latéral

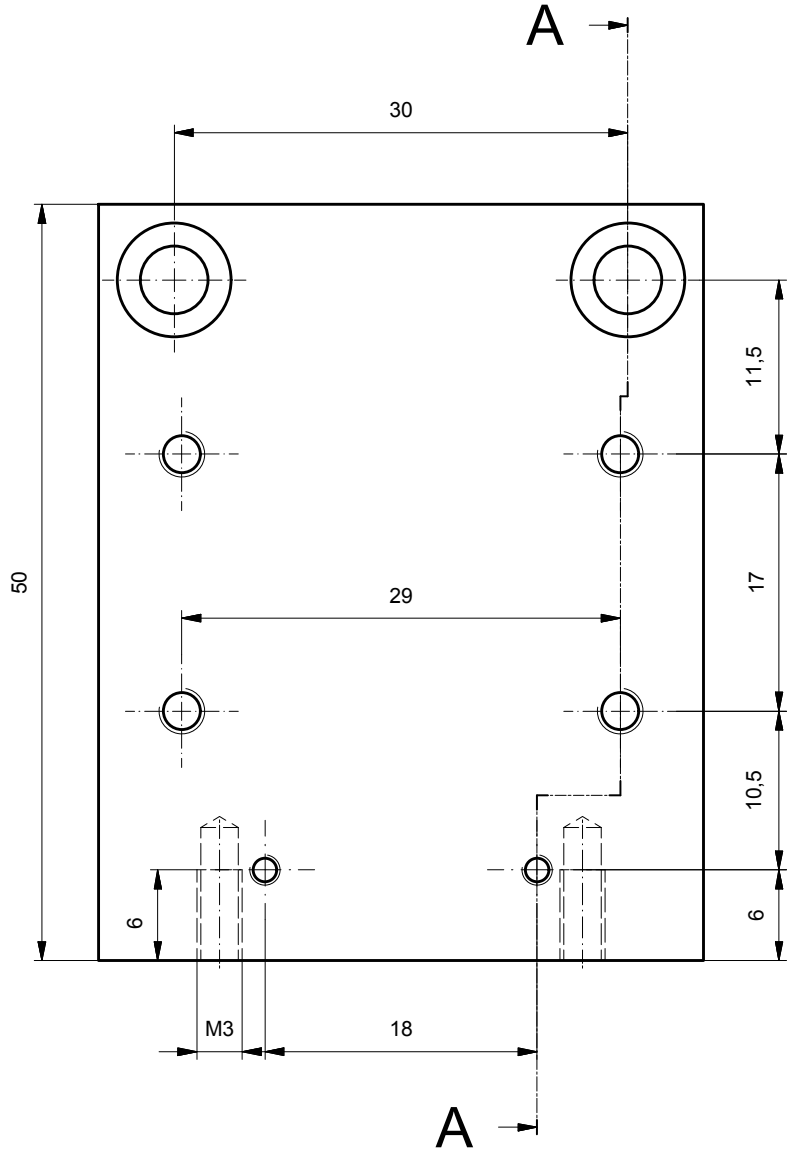
Support latéral	Dessiné Gezeichnet	Vincent Giachino	29.05.2015	Echelle Massstab
Projet BRAAVOO	Contrôlé Geprüft			2:1

Fichier
Datei U:\PRD\Composants\Support de mesure\Support_Lateral.idw

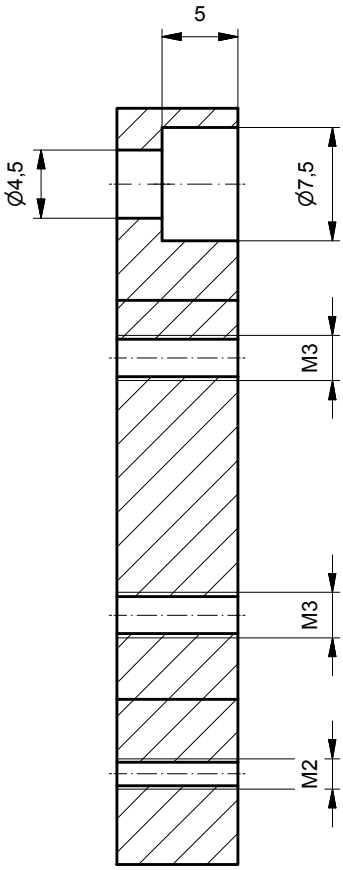
Hes·so VALAIS WALLIS







A-A (2 : 1)



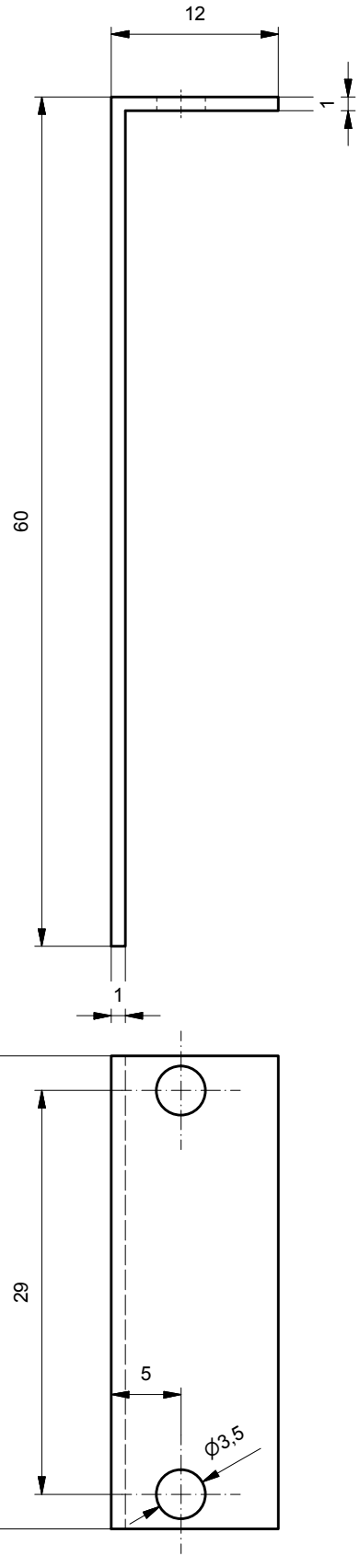
Pos. Pos.	Quantité Menge	Matière Werkstoff	Dénomination / Caractéristiques Benennung / Merkmale
P001	1	Aluminium	Support médian

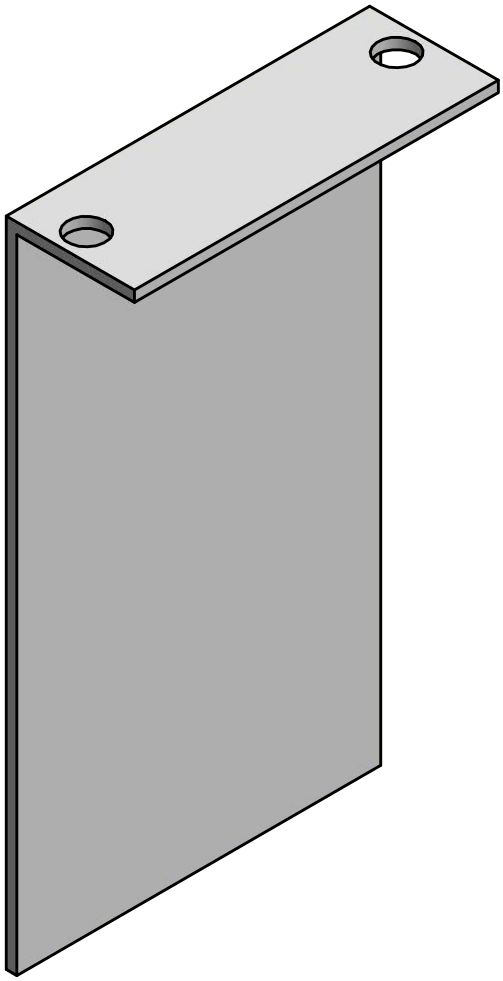
Support médian	Dessiné Gezeichnet	Vincent Giachino	29.05.2015	Echelle Massstab
Projet BRAAVOO	Contrôlé Geprüft			2:1

Fichier
Datei U:\PRD\Composants\Support de mesure\Support_Median.idw

Hes·so VALAIS WALLIS

1



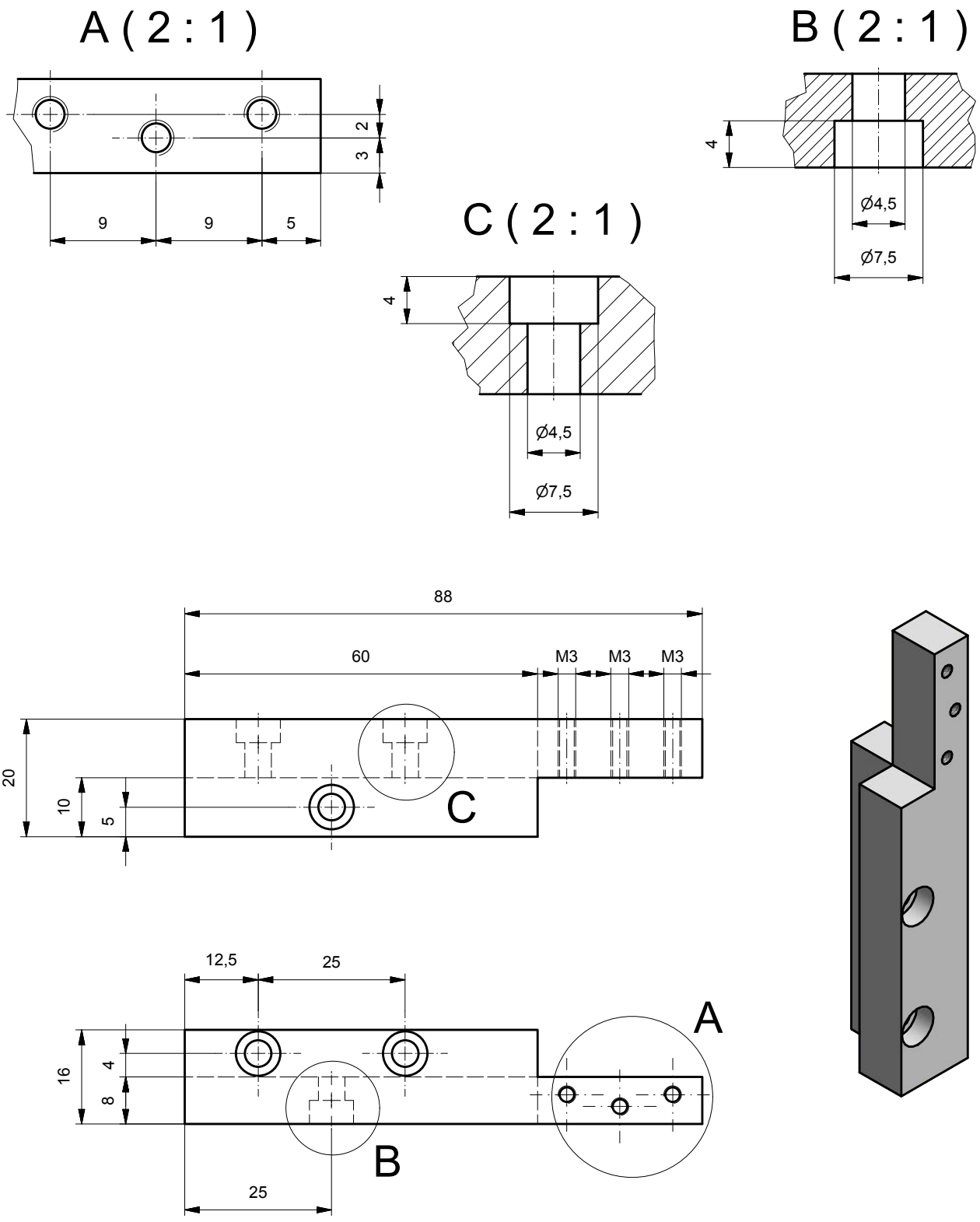


Pos. Pos.	Quantité Menge	Matière Werkstoff	Dénomination / Caractéristiques Benennung / Merkmale
P001	1	Aluminium	Paroi

Paroi Projet BRAAVOO	Dessiné Gezeichnet	Vincent Giachino	16.06.2015	Echelle Massstab
	Contrôlé Geprüft			2:1

Fichier
Datei U:\PRD\Composants\Paroi\Paroi.idw

Hes·so VALAIS WALLIS

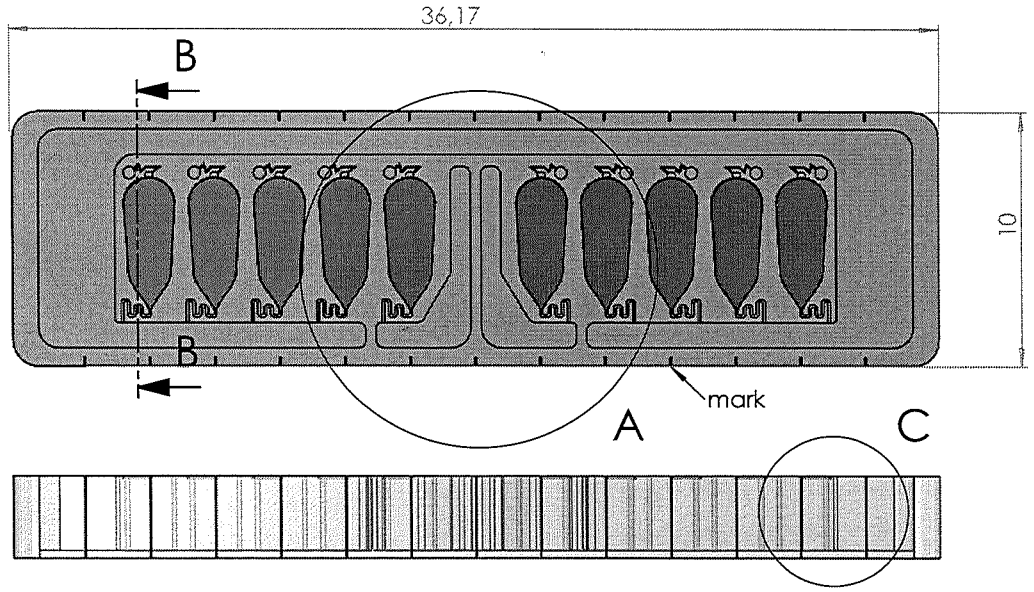


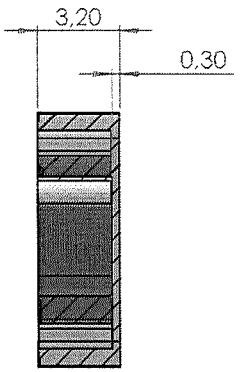
Pos. Pos.	Quantité Menge	Matière Werkstoff	Dénomination / Caractéristiques Benennung / Merkmale
P001	1	Aluminium	Renvoi

Renvoi Projet BRAAVOO	Dessiné Gezeichnet	Vincent Giachino	02.06.2015	Echelle Massstab
	Contrôlé Geprüft			1:1

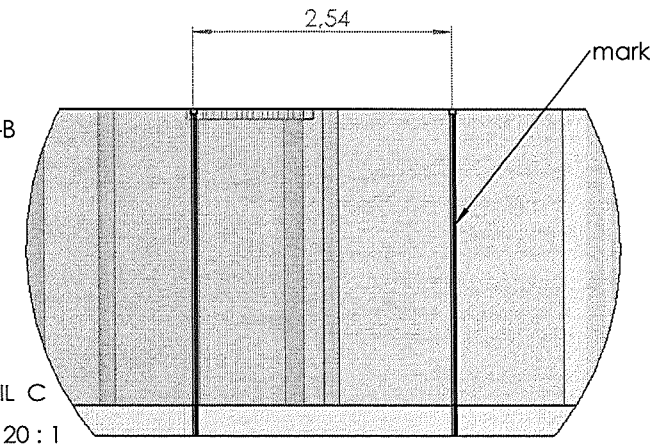
Fichier
Datei U:\PRD\Composants\Support de mesure\Renvoi.idw

Hes·so VALAIS WALLIS



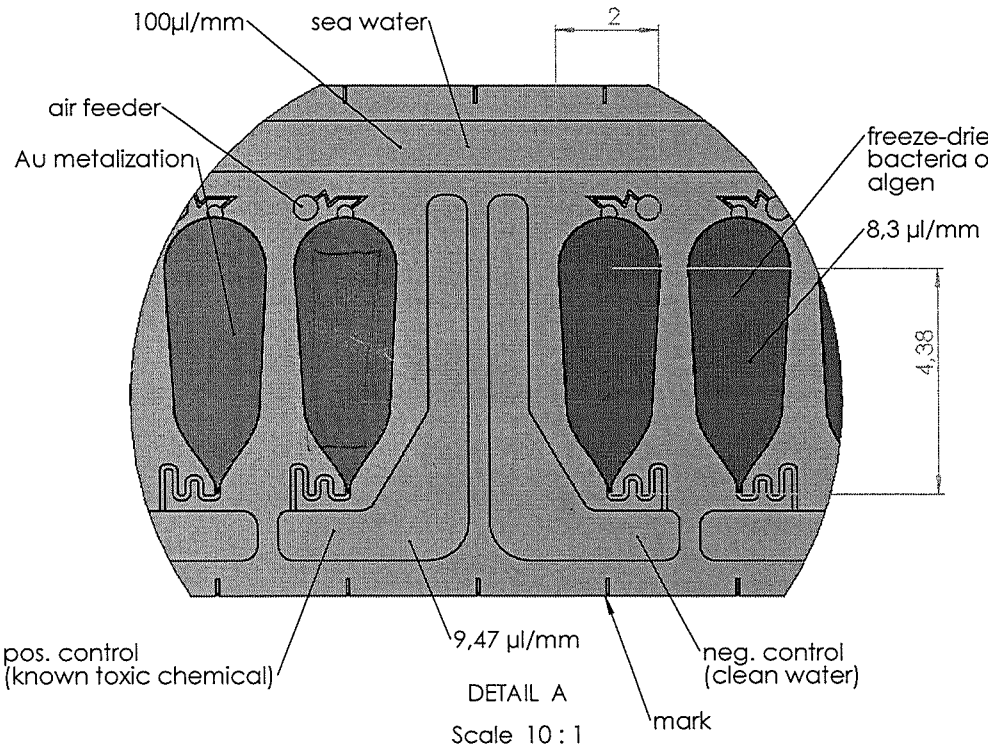


SCHNITT B-B



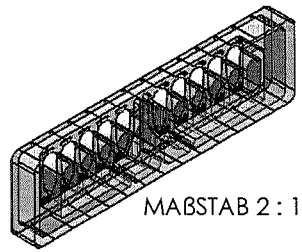
DETAIL C

Scale 20 : 1



DETAIL A

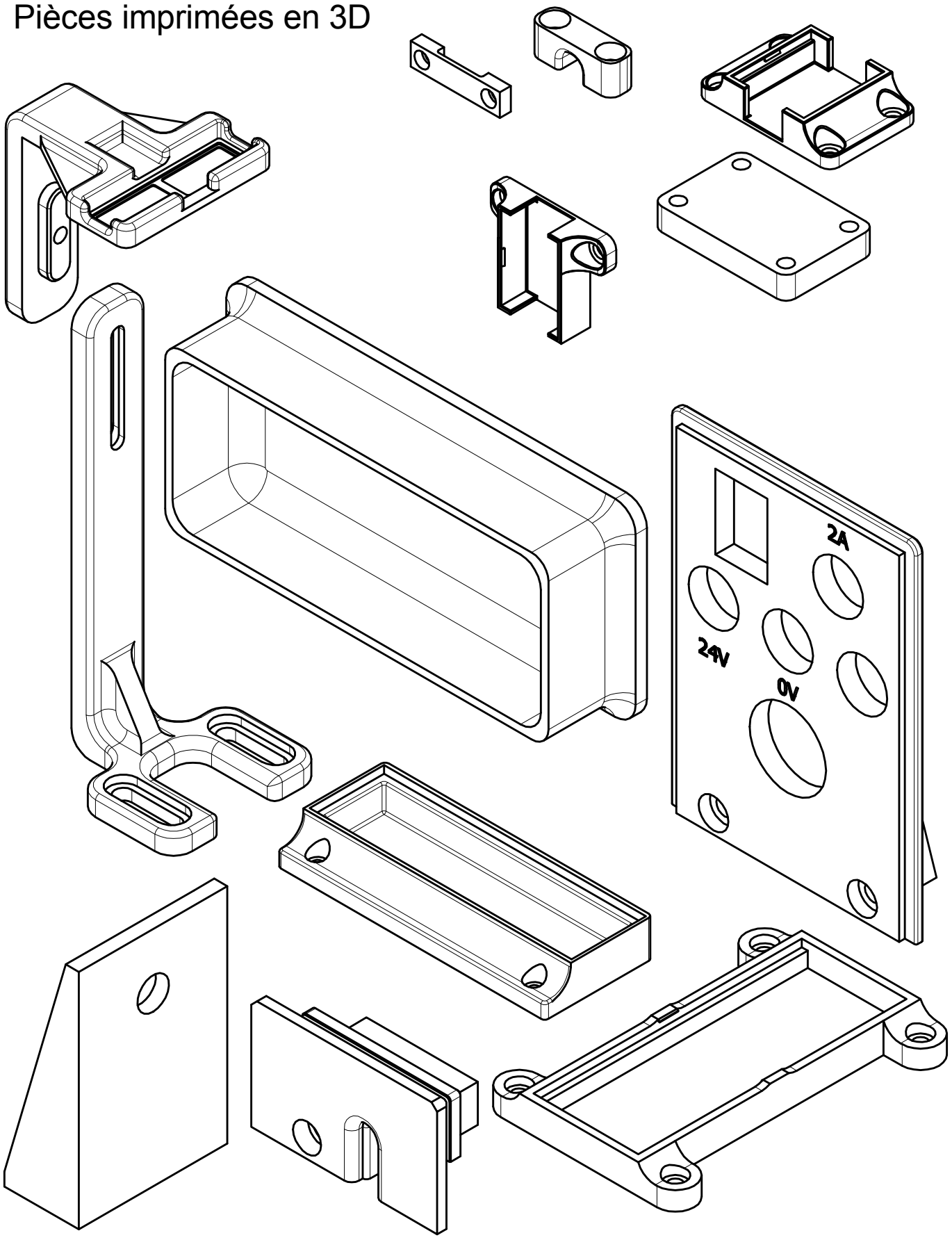
Scale 10 : 1



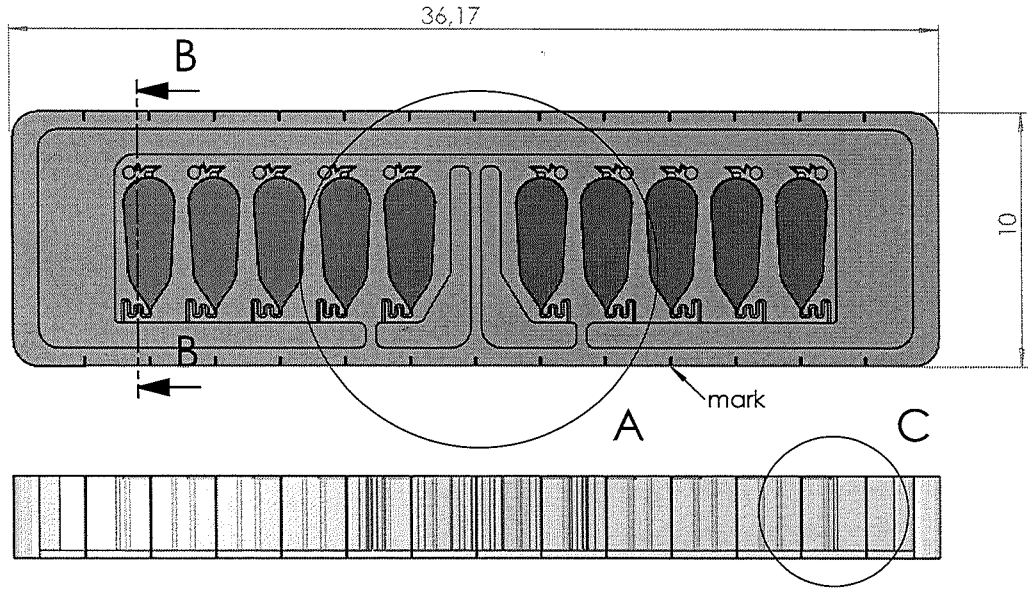
MAßSTAB 2 : 1

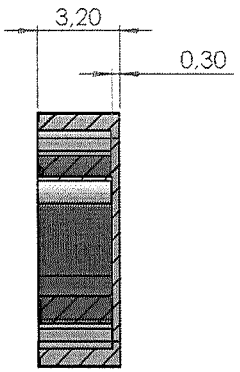
WENN NICHT ANDERS DEFINIERT: BEMASSUNGEN SIND IN MILLIMETER OBERFLÄCHENBESCHAFFENHEIT: TOLERANZEN: LINEAR: WINKEL:	OBERFLÄCHENGÜTE:			ZEICHNUNG NICHT SKALIEREN	ÄNDERUNG
				microTEC Ges. f. Mikrotechnologie mbH	
	NAME	SIGNATUR	DATUM	BENENNUNG:	
GEZEICHNET	Carla		06.11.2014	PAO_f_PMT_10Cavy	
GEPRÜFT					
GENEHMIGT					
PRODUKTION					
QUALITÄT				ZEICHNUNGSNR.	A3
			WERKSTOFF: RMPD Material	13.12.015.12_30	
			GEWICHT:	MASSSTAB:5:1	BLATT 1 VON 1

Pièces imprimées en 3D

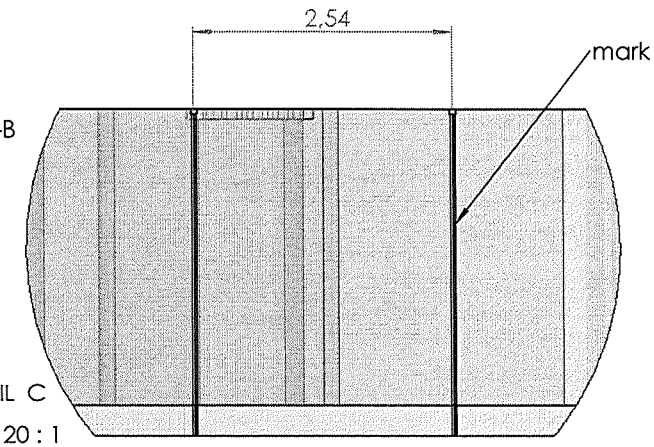


Pièces imprimées en 3D	Dessiné Gezeichnet	Vincent Giachino	07.07.2015	Echelle Massstab
Projet BRAAVOO	Contrôlé Geprüft			1:1
Fichier Datei U:\PRD\Composants\Assemblage\Pieces_Imprimees_3D.idw				
Hes·so VALAIS WALLIS	1			



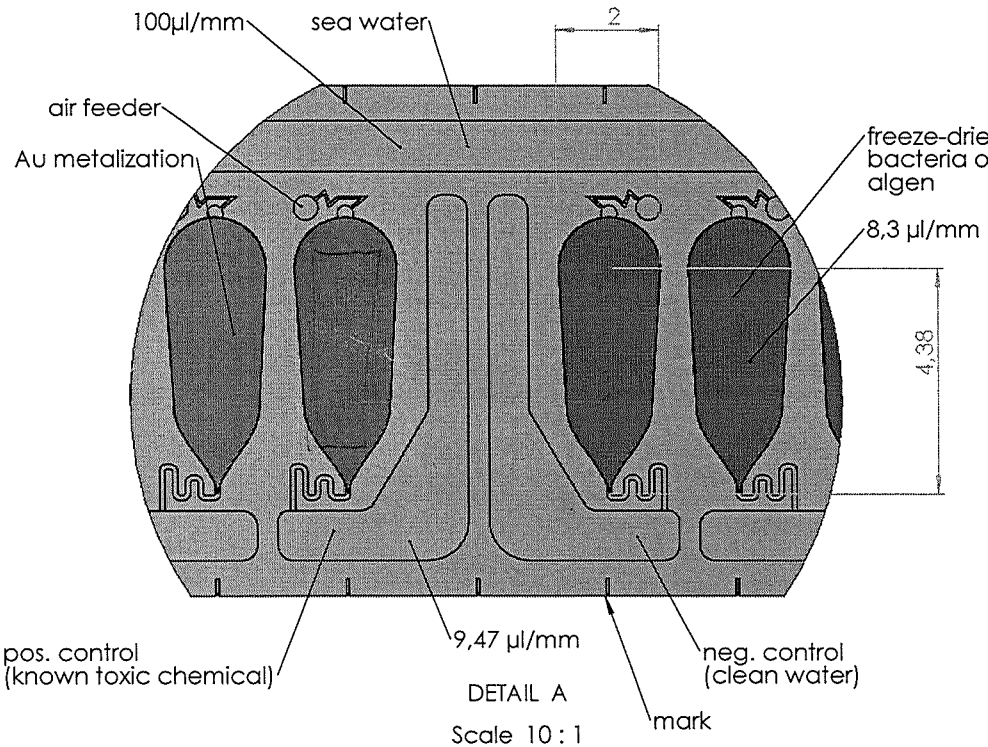


SCHNITT B-B



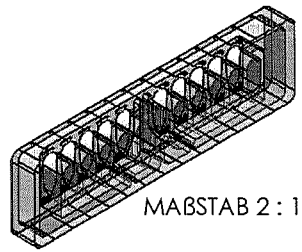
DETAIL C

Scale 20 : 1



DETAIL A

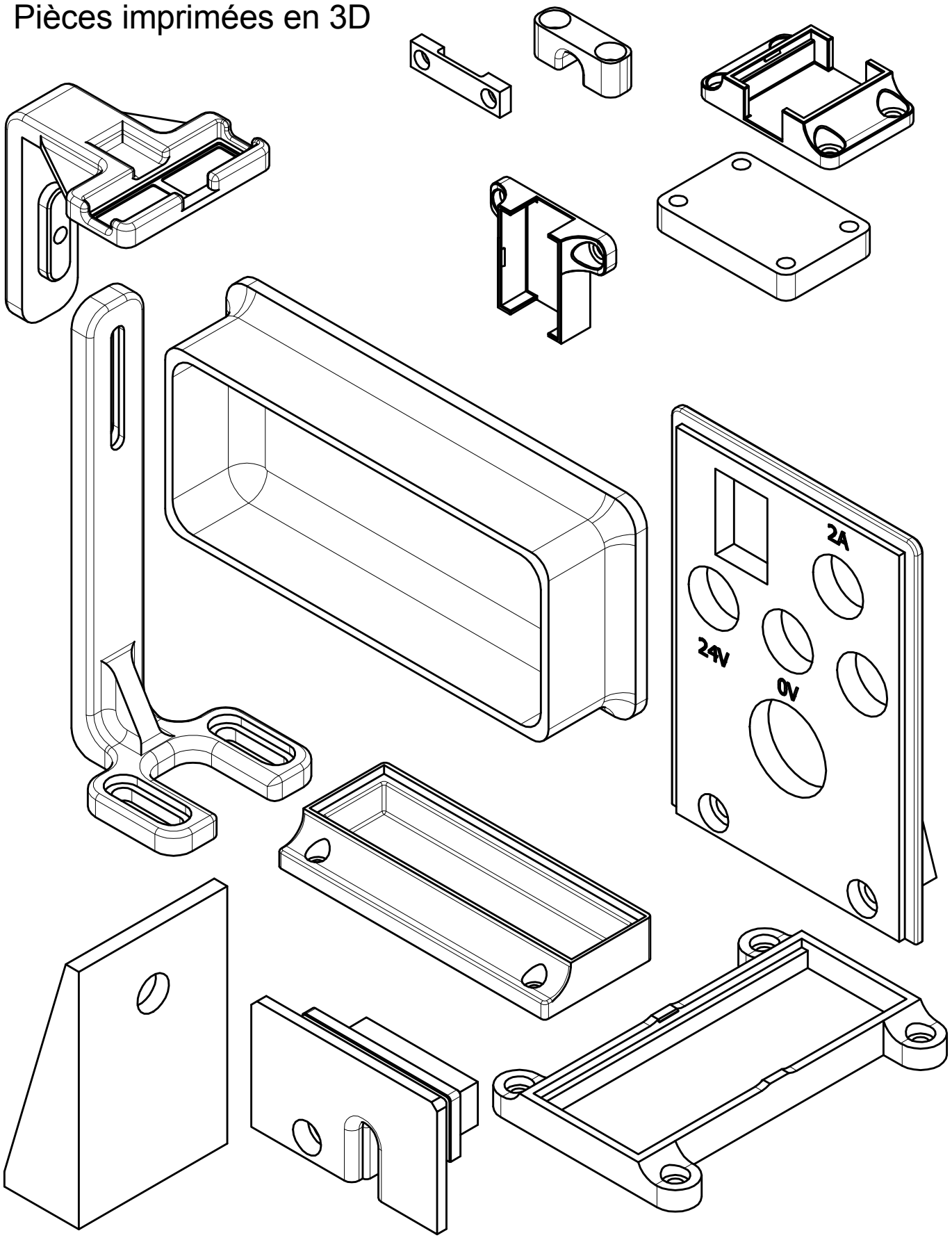
Scale 10 : 1



MAßSTAB 2 : 1

WENN NICHT ANDERS DEFINIERT: BEMASSUNGEN SIND IN MILLIMETER OBERFLÄCHENBESCHAFFENHEIT: TOLERANZEN: LINEAR: WINKEL:	OBERFLÄCHENGÜTE:			ZEICHNUNG NICHT SKALIEREN	ÄNDERUNG
				microTEC Ges. f. Mikrotechnologie mbH	
	NAME	SIGNATUR	DATUM	BENENNUNG:	
GEZEICHNET	Carla		06.11.2014	PAO_f_PMT_10Cavy	
GEPRÜFT					
GENEHMIGT					
PRODUKTION					
QUALITÄT				ZEICHNUNGSNR.	A3
			WERKSTOFF: RMPD Material	13.12.015.12_30	
			GEWICHT:	MASSSTAB:5:1	BLATT 1 VON 1

Pièces imprimées en 3D



Pièces imprimées en 3D	Dessiné Gezeichnet	Vincent Giachino	07.07.2015	Echelle Massstab
Projet BRAAVOO	Contrôlé Geprüft			1:1
Fichier Datei U:\PRD\Composants\Assemblage\Pieces_Imprimees_3D.idw				
Hes·so VALAIS WALLIS	1			

To set up your Raspberry Pi you will need:

	Item	Minimum recommended specification & notes
1	SD card	• Minimum size 4Gb; class 4 (the *class* indicates how fast the card is). • We recommend using branded SD cards as they are more reliable.
2a	HDMI to HDMI / DVI lead	• HDMI to HDMI lead (for HD TVs and monitors with HDMI input). **OR** HDMI to DVI lead (for monitors with DVI input). • Leads and adapters are available for few pounds -- there is no need to buy expensive ones!
2b	RCA video lead	• A standard RCA composite video lead to connect to your analogue display if you are not using the HDMI output.
3	Keyboard and mouse	• Any standard USB keyboard and mouse should work. • Keyboards or mice that take a lot of power from the USB ports, however, may need a powered USB hub. This may include some wireless devices.
4	Ethernet (network) cable [optional]	• Networking is optional, although it makes updating and getting new software for your Raspberry Pi much easier.
5	Power adapter	• A good quality, micro USB power supply that can provide at least **700mA at 5V** is essential. • Many mobile phone chargers are suitable—check the label on the plug. • If your supply provides less than 5V then your Raspberry Pi may not work at all, or it may behave erratically. Be wary of very cheap chargers: some are not what they claim to be. • It does not matter if your supply is rated at *more* than 700mA.
6	Audio lead [optional]	• If you are using HDMI then you will get digital audio via this. • If you are using the analogue RCA connection, stereo audio is available from the 3.5mm jack next to the RCA connector.

Know your leads:

HDMI connector

HDMI to DVI lead

RCA composite video connector

Preparing your SD card for the Raspberry Pi

The SD card contains the Raspberry Pi's operating system (the OS is the software that makes it work, like Windows on a PC or OSX on a Mac). This is very different from most computers and it is what many people find the most daunting part of setting up their Raspberry Pi. It is actually very straightforward—just different!

The following instructions are for Windows users. Linux and Mac users can find instructions at www.raspberrypi.org/downloads

1. Download the Raspberry Pi operating system

The recommended OS is called *Raspbian*. Download it here:
http://downloads.raspberrypi.org/images/raspbian/2012-12-16-wheezy-raspbian/2012-12-16-wheezy-raspbian.zip

2. Unzip the file that you just downloaded

a) Right click on the file and choose "Extract all".
b) Follow the instructions—you will end up with a file ending in *.img*

This *.img* file can only be written to your SD card by special disk imaging software, so…

3. Download the Win32DiskImager software

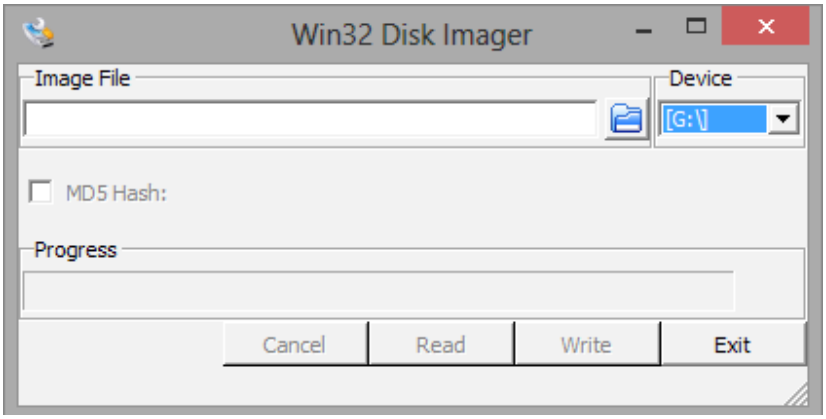
a) Download *win32diskimager-binary.zip* (currently version 0.6) from:
https://launchpad.net/win32-image-writer/+download

b) Unzip it in the same way you did the Raspbian *.zip* file
c) You now have a new folder called *win32diskimager-binary*

You are now ready to write the Raspbian image to your SD card.

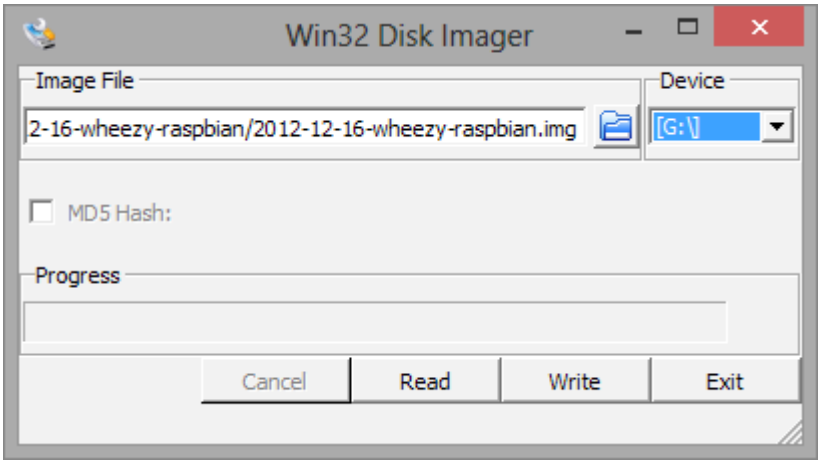
4. Writing Raspbian to the SD card

a) Plug your SD card into your PC
b) In the folder you made in step 3(b), run the file named *Win32DiskImager.exe* (in Windows Vista, 7 and 8 we recommend that you right-click this file and choose "Run as administrator"). You will see something like this:



c) If the SD card (*Device*) you are using isn't found automatically then click on the drop down box and select it

d) In the *Image File* box, choose the Raspbian *.img* file that you downloaded



e) Click *Write*
f) After a few minutes you will have an SD card that you can use in your Raspberry Pi

5. Booting your Raspberry Pi for the first time

a) Follow the *Quick start* guide on page 1

b) On first boot you will come to the `Raspi-config` window

c) Change settings such as `timezone` and `locale` if you want

d) Finally, select the second choice:
`expand_rootfs`

and say 'yes' to a reboot

e) The Raspberry Pi will reboot and you will see **`raspberrypi login:`**

f) Type:
`pi`

g) You will be asked for your **Password**

h) Type:
`raspberry`

i) You will then see the prompt:
`pi@raspberry ~ $`

j) Start the desktop by typing:
`startx`

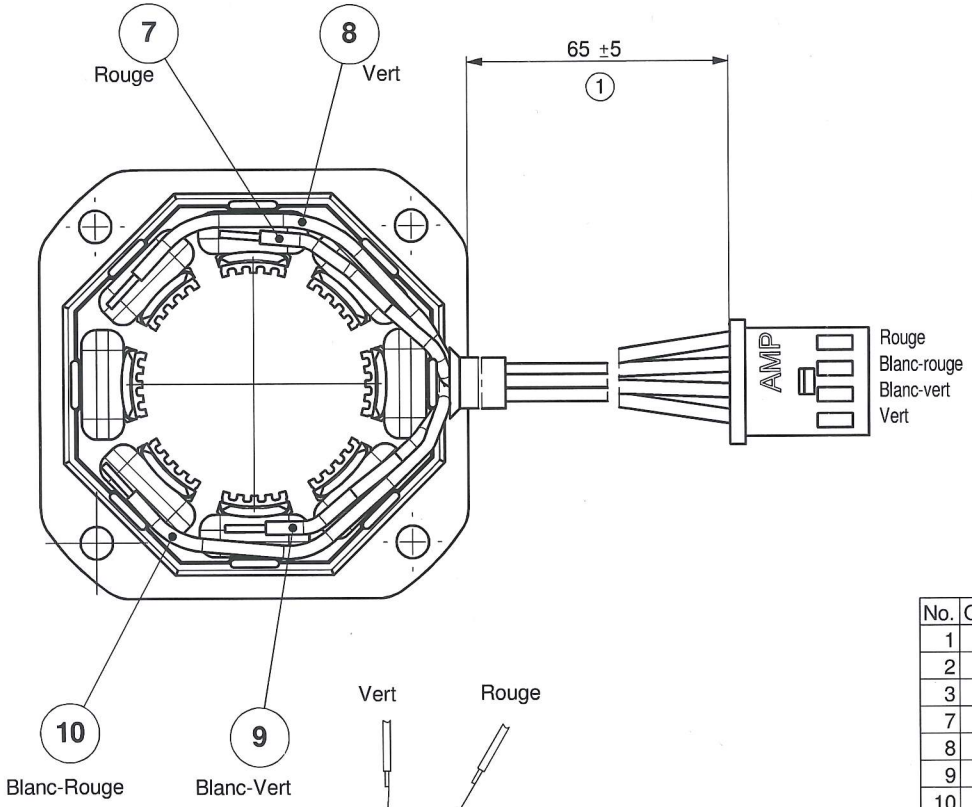
k) You will find yourself in a familiar-but-different desktop environment.

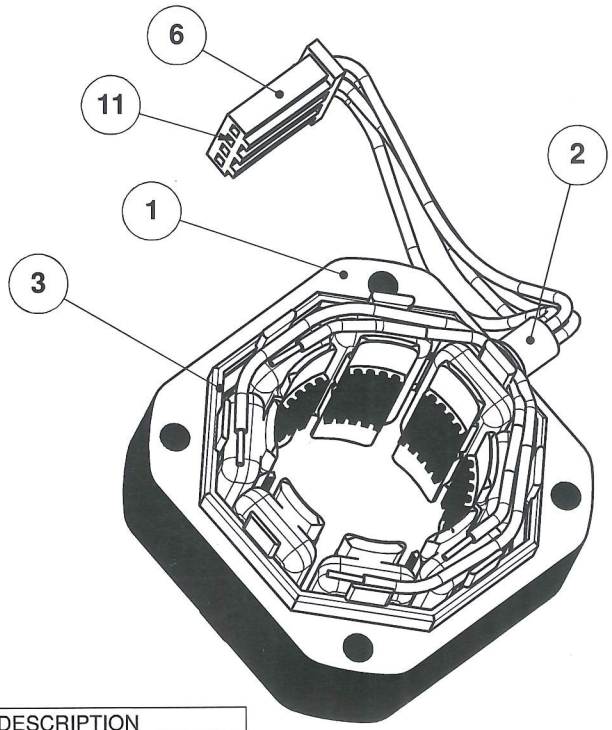
l) Experiment, explore and have fun!

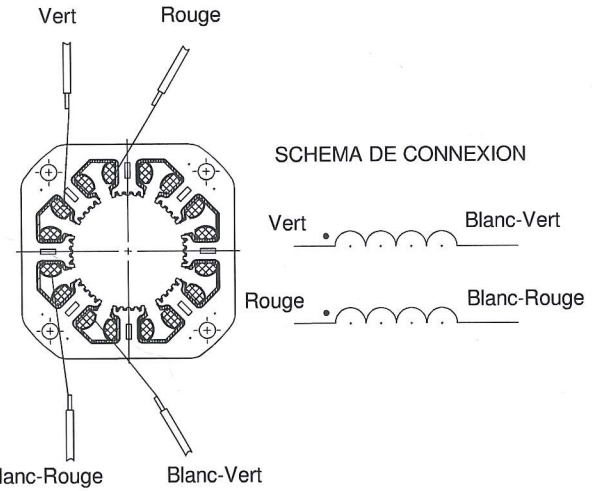
For more details and where to go next visit www.raspberrypi.org and the forums at www.raspberrypi.org/phpBB3

The latest version of Raspbian can always be found at www.raspberrypi.org/downloads

I Phase A	Résistance Ohm	Spires	Diamètre fil	Poids g	Qualité	N° normalisé
0.34	22	8x202	0.2	21	Cu C1.3 incolore	FB13D002000







SCHEMA DE CONNEXION

No.	QTÉ.	No.PIÈCE	DESCRIPTION
1	1	6540G114	PAQUET STATOR
2	1	6540P012	PASSE-FILS
3	2	6540P020	PIECE TERMINALE
7	1	6541P120	CABLE ROUGE
8	1	6541P121	CABLE VERT
9	1	6541P123	CABLE BLANC-VERT
10	1	6541P122	CABLE BLANC-ROUGE
	*	6540P036	COLLE
11	4	6110P140	CONTACT
6	1	6110P170	AMP 280359-0

Matière :

Traitement thermique :

Fini :

Code mat. int.:

Modification 02.11.2004 amj

TOLERANCES GENERALES (mm) ISO 2768-m

≦ 6	> 6 à 30	> 30 à 120	> 120 à 400
± 0.1	± 0.2	± 0.3	± 0.5

Distribution centre: 520

STATOR BOBINE

Echelle 2:1

Remplacé par:

Remplace:

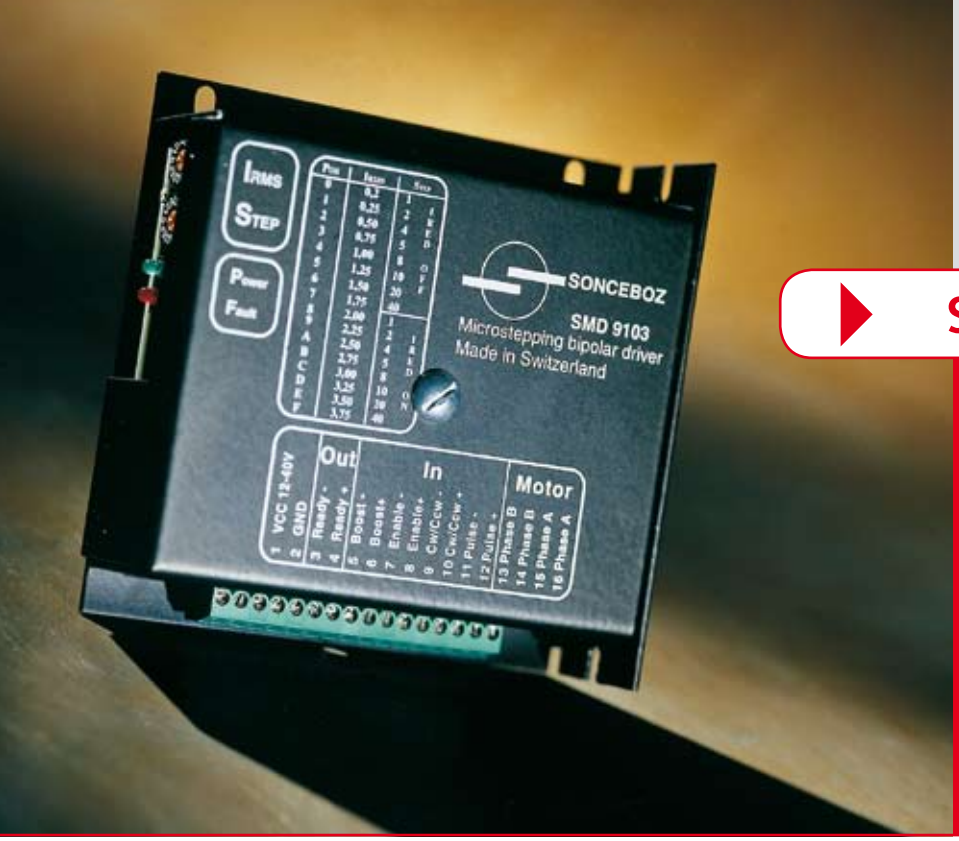
Dessiné	16.02.04	CDX
Contrôlé	16.02.04	TM

SONCEBOZ

SONCEBOZ SA
CH-2605 Sonceboz

6540G612.001

A3



SONCEBOZ

Stepper motor controller

9103

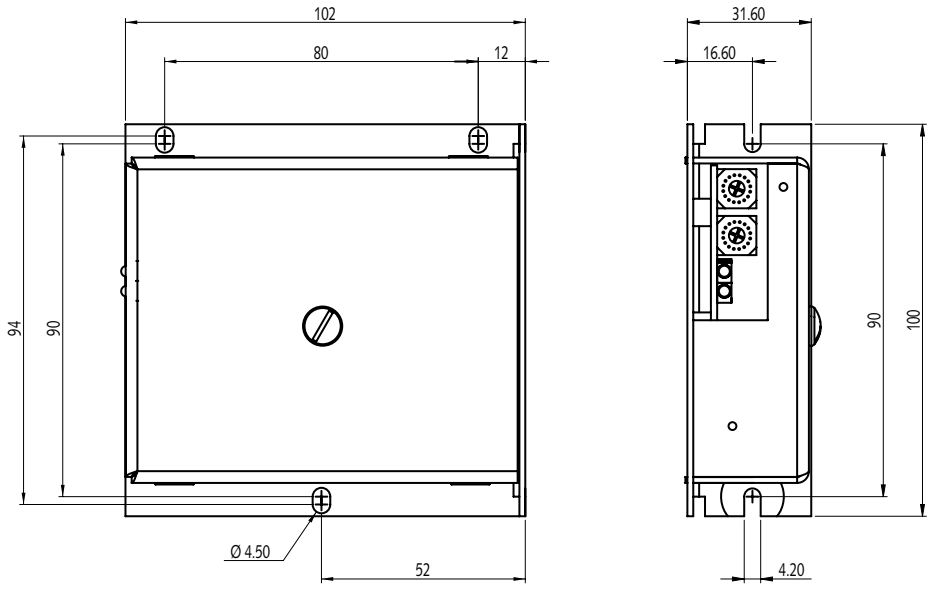
RoHS compliant 2002 / 95 / EC

- Adjustable Microstep ..... 1/2 to 1/40
- Adjustable Motor Current ... 0.2 to 4 Aeff
- Interface ..... Pulse
  Direction
  Boost
  Enable
  Ready (output)
- Dimensions ..... 100x102x32 mm

- **Compact:** the integration of a powerful microcontroller offers an user-friendly driver with a high degree of flexibility.
- **Interface:** all digital inputs have optocouplers (isolation). Microsteps and motor current are adjustable when the driver is running.
- **High step input frequency:** the driver allows a step frequency up to 200 kHz.
- **High resolution:** the controller drives motor, with 200 steps per revolution, to rotate practically without vibration and with a resolution from 1.8° (full step) to 0.045° (1/40).
- **Intelligent driver:** enable the boost input increases the torque up to 33 % and is particularly useful for acceleration. In order to minimize the temperature rise, the current is automatically reduced to 2/3 of the nominal value at standstill.

Dimensions

Drawing not to scale. All dimensions in mm.



Technical data

Designation	Value	Unit
Nominal phase current	3.75	A
Max current per phase with Boost	4	A
Motor current selection (Inom)	0.2 ; 0.25 ... 3.75	A
Increment	0.25	A
Boosted current (limited to 4 A max)	133% Inom	-
Reduced current (limited to 0.2 A min)	66% Inom	-
Reduced current delay after last step only available with step selection from 8 to F	100	ms
Power supply: Inom ≤ 2.5 A Inom > 2.5 A	12 ... 40 18 ... 40	V V
Selectable microstep	1/1 ; 1/2 ; 1/4 ; 1/5 ; 1/8 ; 1/10 ; 1/20 ; 1/40	-
Chopper frequency	32	kHz
Maximum step input frequency	200	kHz
Optocoupler inputs for: - motor current on (Enable) - step pulses (Pulse) - direction of rotation (CW/CCW) - motor current boost (Boost)	Typical: 5 16	 V mA
Optocoupler output for: - No default (Ready)	Max: 30 10	 V mA

Error detection and diagnosis

State	Ready output	Green LED	Red LED	Current	Restart
No error	1	ON	OFF	Enabled	-
Motor short-circuit	0	OFF	ON	Disabled	No
Maximum step input frequency	0	ON	Blinking	Disabled	Enable = 0
Overtemperature (> 80° C)	0	Blinking	Blinking	Disabled	T < 75° C and Enable = 0
Power supply undervoltage *< 16 Vdc, **< 11 Vdc	0	Blinking	OFF	Disabled	*>18 Vdc or **>12 Vdc and Enable = 0
Pwer supply overvoltage (> 45 Vdc)	0	Blinking	ON	Disabled	< 40 Vdc and Enable = 0

*≤ 2.5 A ** > 2.5 A

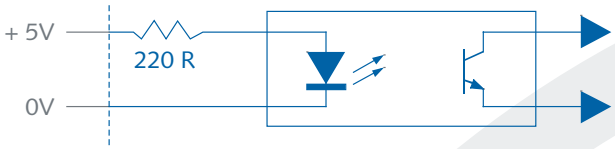
SONCEBOZ SA
2605 Sonceboz - Switzerland
Tel. +41(0) 32 488 11 11
Fax +41(0) 32 488 11 00
E-mail : info@sonceboz.com
Internet : www.sonceboz.com

SONCEBOZ

Electrical Interface

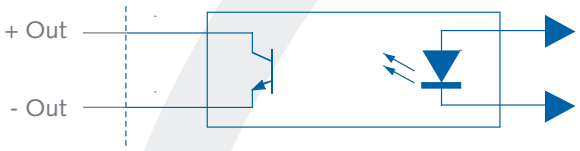
• Inputs

The inputs are compatible with 5 V logic. The increase of logical voltage is allowable up to 24 Vdc when an external resistor is added in order to respect the typical value of the input current.



• Output

The Ready output has an optocoupler link (Open collector). The maximum value is 30 Vdc for the supply and 10 mA for the current.



• Connector pin assignment

Pin N°	Signal	Description
1	VCC +	Supply 12 (18) – 40 Vdc
2	GND	
3	Ready -	No default output
4	Ready +	
5	Boost - (0 V)	Motor current boost input
6	Boost + (5 V)	
7	Enable - (0 V)	Motor current on input
8	Enable + (5 V)	
9	CW/CCW - (0 V)	Direction of rotation input
10	CW/CCW + (5 V)	
11	Pulse - (0 V)	Step pulse input
12	Pulse + (5V)	
13	Phase B2	Phase B of motor
14	Phase B1	
15	Phase A2	Phase A of motor
16	Phase A1	

Directives & Operating conditions

Operating temperature, Inom ≤ 2 A	0… 60° C
Operating temperature, Inom > 2 A	0… 40° C
Storage temperature	-40… + 125° C
Directives	2002/95/EC (RoHS)

Installation

The cables insulation must withstand at mean 85° C. The cables cross-section must be at mean AWG. The cables between the power supply and the controller, and between the controller and the motor must not be longer than 0.3 m.

Ordering information

Type	Ordering code
9103 standard	9103R002

SONCEBOZ SA
2605 Sonceboz - Switzerland
Tel. +41(0) 32 488 11 11
Fax +41(0) 32 488 11 00
E-mail : info@sonceboz.com
Internet : www.sonceboz.com

SONCEBOZ

DC/DC-Konverter

TEN 10 Serie, 10 Watt

UL 60950-1

Merkmale

- Weite 2:1 Eingangsbereiche
- Hohe Leistungsdichte
- Arbeitstemperaturbereich –40 °C bis +85 °C
- Dauerkurzschlussfest
- E/A-Isolation 1500 VDC
- Eingangsfilter nach EN 55022, Klasse A und FCC, Level A ohne externe Komponenten
- Industriestandard-Pinning
- Abgeschirmtes Metallgehäuse mit isolierter Bodenplatte
- Hohe Zuverlässigkeit, MTTF > 1 Mio. Std.
- 3 Jahre Produktgewährleistung

Die TEN 10 Serie bietet hochqualitative 10 W DC/DC-Konverter in kompakter 50.8 x 25.4 mm Bauform mit Industriestandard-Pinning. Der hohe Wirkungsgrad ermöglicht einen erweiterten Arbeitstemperaturbereich von –40°C bis +85°C. Weitere Merkmale dieser Serie sind das integrierte Filter nach EN 55022, Klasse A ohne externe Komponenten, Überspannungsschutz und die Dauerkurzschlussfestigkeit. Typische Anwendungen für diese Serie liegen im Bereich mobiler batterieversorgter Stromversorgungen in Industrie- und Kommunikationssystemen, kurzum überall dort wo galvanisch getrennte, genau regulierte Spannungen erforderlich sind.

Modelle

Bestellnummer	Eingangsspannung	Ausgangsspannung	Ausgangsstrom max.	Wirkungsgrad typ.
TEN 10-1210	9 – 18 VDC (12 VDC Nominal)	3.3 VDC	2'400 mA	72 %
TEN 10-1211		5 VDC	2'000 mA	77 %
TEN 10-1212		12 VDC	830 mA	80 %
TEN 10-1213		15 VDC	670 mA	80 %
TEN 10-1215		24 VDC	415 mA	81 %
TEN 10-1221		±5 VDC	±1'000 mA	78 %
TEN 10-1222		±12 VDC	±415 mA	81 %
TEN 10-1223		±15 VDC	±330 mA	80 %
TEN 10-2410	18 – 36 VDC (24 VDC Nominal)	3.3 VDC	2'400 mA	76 %
TEN 10-2411		5 VDC	2'000 mA	78 %
TEN 10-2412		12 VDC	830 mA	82 %
TEN 10-2413		15 VDC	670 mA	82 %
TEN 10-2415		24 VDC	415 mA	83 %
TEN 10-2421		±5 VDC	±1'000 mA	80 %
TEN 10-2422		±12 VDC	±415 mA	82 %
TEN 10-2423		±15 VDC	±330 mA	82 %
TEN 10-4810	36 – 75 VDC (48 VDC Nominal)	3.3 VDC	2'400 mA	76 %
TEN 10-4811		5 VDC	2'000 mA	80 %
TEN 10-4812		12 VDC	830 mA	82 %
TEN 10-4813		15 VDC	670 mA	83 %
TEN 10-4815		24 VDC	415 mA	83 %
TEN 10-4821		±5 VDC	±1'000 mA	81 %
TEN 10-4822		±12 VDC	±415 mA	83 %
TEN 10-4823		±15 VDC	±330 mA	83 %

Eingangsspezifikationen

Eingangsstrom bei Leerlauf	12 Uein Modelle:	30 mA typ.
	24 Uein Modelle:	20 mA typ.
	48 Uein Modelle:	10 mA typ.
Eingangsstrom bei Vollast	12 Uein; 3.3 VDC Uaus Modell:	915 mA typ.
	12 Uein; 5 & ±5 VDC Modelle:	1080 mA typ.
	12 Uein; andere Modelle:	1045 mA typ.
	24 Uein; 3.3 VDC Modell:	435 mA typ.
	24 Uein; 5 & ±5 VDC Modelle:	530 mA typ.
	24 Uein; andere Modelle:	510 mA typ.
	48 Uein; 3.3 VDC Modell:	215 mA typ.
	48 Uein; 5 & ±5 VDC Modelle:	260 mA typ.
	48 Uein; andere Modelle:	250 mA typ.
Startspannung / Unterspannungsabschaltung	12 Uein Modelle:	8.5 VDC / 8 VDC
	24 Uein Modelle:	16.5 VDC / 16 VDC
	48 Uein Modelle:	32.5 VDC / 32 VDC
Transiente Überspannung (1 sec. max.)	12 Uein Modelle:	25 V max.
	24 Uein Modelle:	50 V max.
	48 Uein Modelle:	100 V max.
Verpolungsschutz		1.0 A max.
Leitungsgebundene Störungen (Eingang)		EN 55022 Klasse A, FCC Teil 15, Level A

Ausgangsspezifikationen

Einstellbereich der Ausgangsspannung		±1 % max
Regelabweichungen	– Eingangsänderung Uein min. bis Uein max.	0.3 % max.
	– Laständerung 10 – 100 % Singlemodelle:	0.5 % max.
	Dualmodelle:	1.0 % max. (symmetrische Last)
	Dualmodelle:	3.0 % max. (unsymmetrische Last)
Restwelligkeit (20 MHz Bandbreite)	Singlemodelle:	50 mVpk-pk typ.
	Dualmodelle:	75 mVpk-pk typ.
Temperaturkoeffizient		±0.02 %/K
Strombegrenzung		> 110 % Iaus max., Konstantstrom
Kurzschlußschutz		dauernd, automatischer Neustart
Kapazitive Last	Singlemodelle:	2200 µF max.
	Dualmodelle:	470 µF max.

Allgemeine Spezifikationen

Temperaturbereich	– Betrieb	–40 °C bis +85 °C
	– Gehäusetemperatur	+100 °C max.
	– Lagerung	–40 °C bis +125°C
Leistungsreduktion (Konvektionskühlung)		3.3 %/K oberhalb 70 °C
Luftfeuchtigkeit (nicht betauend)		95 % rel H max.
Zuverlässigkeit, kalkulierte MTBF (MIL-HDBK-217F, +25 °C, ground benign)		> 1 Mio. Std.
Isolation (Eingang/Ausgangt)	– Spannung	1500 VDC
	– Kapazität	120 pF max.
	– Widerstand	> 1 GΩ
Schaltfrequenz	Singlemodelle:	500 kHz typ. (Pulsbreitenmodulation)
	Dualmodelle:	300 kHz typ. (Pulsbreitenmodulation)

Alle Spezifikationen bei Nominal-Eingangsspannung, Vollast und +25 °C nach Aufwärmzeit, ausgenommen anders spezifiziert.

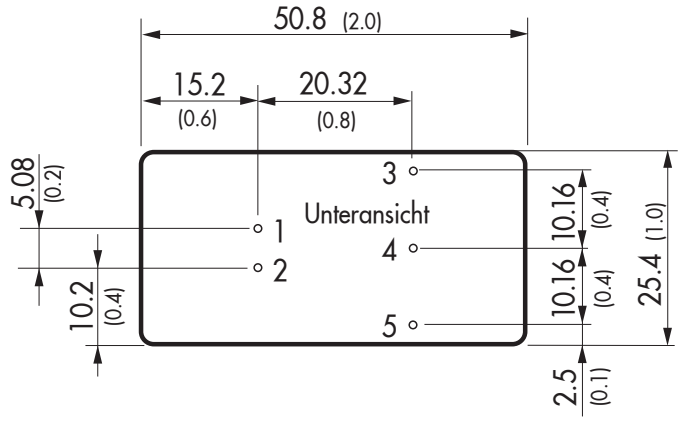
Allgemeine Spezifikationen

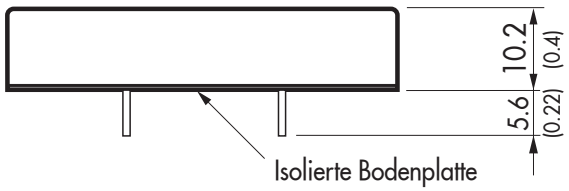
Elektromagnetische Verträglichkeit (EMV), Störfestigkeit	– Elektrostatische Entladung ESD	EN 61000-4-2	8 kV / 6 kV, Kriterium B
	– Elektromagnetische Einstrahlung HF	EN 61000-4-3	10 V/m, Kriterium A
	– Schnelle Transienten / Bursts auf Eingangsltg.	EN 61000-4-4	±2 kV, Kriterium B
	– Surge / Blitzimpuls	EN 61000-4-5	±1 kV, Kriterium B
	– HF-Einkopplungen auf Eingangsltg.	EN 61000-4-6	10 $V_{eff.}$, Kriterium A
Vibration		MIL-STD-810F	
Thermischer Schock		MIL-STD-810F	
Sicherheitsstandards		UL 60950-1, IEC / EN 60950-1	
Sicherheitszulassungen	– UL/cUL	www.ul.com -> Zertifikate -> File Nr. E188913	

Physikalische Spezifikationen

Gehäusematerial	Stahl, vernickelt
Bodenplatte	Epoxid
Vergussmasse	Silikon TES (UL 94 V-0 Klasse)
Gewicht	30 g
Löttemperatur	max. 265 °C / 10 sec.

Gehäuseabmessungen





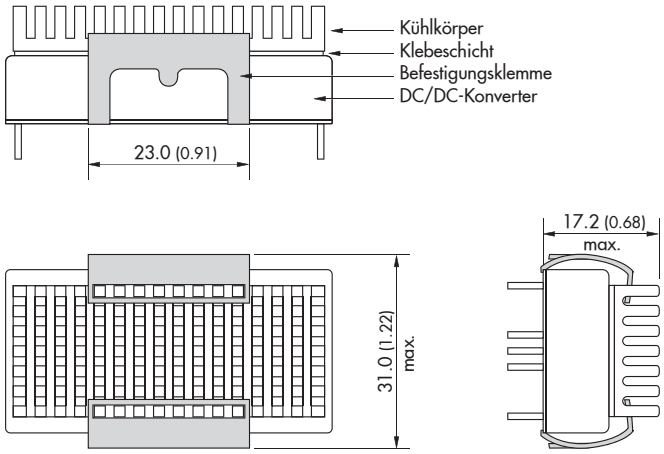
Pin-Out		
Pin	**Single**	**Dual**
1	+ Uein (Vcc)	+ Uein (Vcc)
2	– Uein (GND)	– Uein (GND)
3	+ Uaus	+ Uaus
4	Kein Pin	Common
5	– Uaus	– Uaus

Abmessungen in [mm], () = Inch
Pin-Durchmesser: 1.0 ±0.05 (0.02 ±0.002)
Toleranzen-Rastergrundmass: ±0.25 (±0.01)
Gehäuse Toleranzen: ±0.5 (±0.02)

Alle Spezifikationen bei Nominal-Eingangsspannung, Vollast und +25 °C nach Aufwärmzeit, ausgenommen anders spezifiziert.

Kühlkörper (Option)

Kühlkörper TEN-HS4 (Option)



Bestellnummer: **TEN-HS4**
(Enthält: Kühlkörper, Klebeschicht und zwei Befestigungsklemmen).
Material: Aluminum
Oberfläche: Eloxiert (schwarz)
Gewicht: 17 g (ohne Konverter)
Thermischer Widerstand vor Montage: 10 K/W

Anmerkung:
Der Produktaufkleber des DC/DC-Konverters muss vor der Montage des Kühlkörpers entfernt werden.

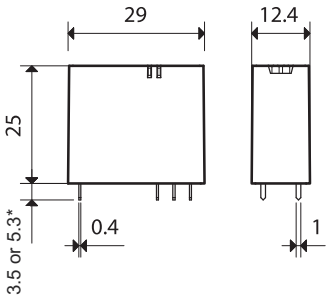
Bei sehr großen Stückzahlen kann der Konverter ab Werk, mit vormontiertem Kühlkörper geliefern werden. Einzelne Kühlkörper sind für Prototypen und kleinere Stückzahlen verfügbar.

TRACO ELECTRONIC GmbH
Oskar-Messter-Strasse 20a • D-85737 Ismaning/München
Tel. +49 89/96 11 82 0 • Fax +49 89/96 11 82 20 • info@traco-electronic.de • www.traco-electronic.de

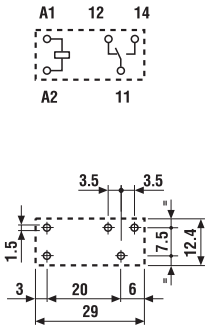
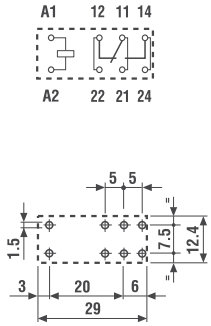
Features

1 Pole relay range
40.31 - 1 Pole 12 A (3.5 mm pin pitch)
40.61 - 1 Pole 16 A (5 mm pin pitch)

- Pin length 3.5 mm for pcb mount
- Pin length 5.3 mm as Plug-in relay
- DC sensitive coils as standard
- Cadmium Free contact material available
- 6 kV (1.2/50 µs) isolation coil-contacts
- 8 mm creepage and clearance distances between coil and contacts
- Meets EN 60335-1 glow wire requirements
- Flux proof: RT II standard or wash tight RT III
- AC inductive load rating (related to AC15 utilisation category) 4 A 250 V approved according to EN 61810-1:2008 (Annex B tables B1, B2, B3)



* (3.5 or 5.3) mm see ordering code

	40.31	40.61
		
	• 3.5 mm contact pin pitch • 1 Pole 12 A	• 5 mm contact pin pitch • 1 Pole 16 A
	 Copper side view	 Copper side view
Contact specification		
Contact configuration	1 CO (SPDT)	1 CO (SPDT)
Rated current/Maximum peak current A	12/20	16/30
Rated voltage/Maximum switching voltage V AC	250/400	250/400
Rated load AC1 VA	3,000	4,000
Rated load AC15 (230 V AC) VA	1,000	1,000
Single phase motor rating (230 V AC) kW	0.55	0.55
Breaking capacity DC1: 30/110/220 V A	12/0.3/0.12	16/0.3/0.12
Minimum switching load mW (V/mA)	300 (5/5)	500 (10/5)
Standard contact material	AgNi	AgCdO
Coil specification		
Nominal voltage (U_N) V AC (50/60 Hz)	–	–
V DC	12 - 24	12 - 24
Rated power W	0.5	0.5
Operating range AC	–	–
DC	(0.73...1.5)U_N	(0.8...1.5)U_N
Holding voltage DC	0.4 U_N	0.4 U_N
Must drop-out voltage DC	0.1 U_N	0.1 U_N
Technical data		
Mechanical life AC/DC cycles	$10 \cdot 10^6$	$10 \cdot 10^6$
Electrical life at rated load AC1 cycles	$200 \cdot 10^3$	$100 \cdot 10^3$
Operate/release time ms	10/3	10/3
Insulation between coil and contacts (1.2/50 µs) kV	6 (8 mm)	6 (8 mm)
Dielectric strength between open contacts V AC	1,000	1,000
Ambient temperature range °C	–40...+85	–40...+85
Environmental protection	RT II	RT II
Approvals (according to type)		RINA

Ordering information

Example: 40 series PCB relay, 1 CO (SPDT) - 12 A, 24 V DC coil.

								A	B	C	D		
4	0	.	3	1	.	7	.	0	2	4	.	1	0	2	0

Series

Type
3 = PCB - 3.5 mm pinning
6 = PCB - 5 mm pinning

No. of poles
1 = 1 pole
for: 40.31, 12 A
40.61, 16 A

Coil version
7 = Sensitive DC

Coil voltage
012 = 12 V DC
024 = 24 V DC

A: Contact material
0 = AgNi
(40.31 Plug-in relays)
0 = AgCdO
(40.61 Plug-in relays)
1 = AgNi (PCB relays)
2 = AgCdO
(40.61 PCB relays)

B: Contact circuit
0 = CO (SPDT)
3 = NO (SPST)

D: Special versions
0 = Standard flux proof (RT II)
1 = Wash tight (RT III)

C: Options
0 = Pins length 5.3 mm (Plug-in relays)
2 = Pins length 3.5 mm (PCB relays)

Selecting features and options: only combinations in the same row are possible.
Preferred selections for best availability are shown in **bold.**

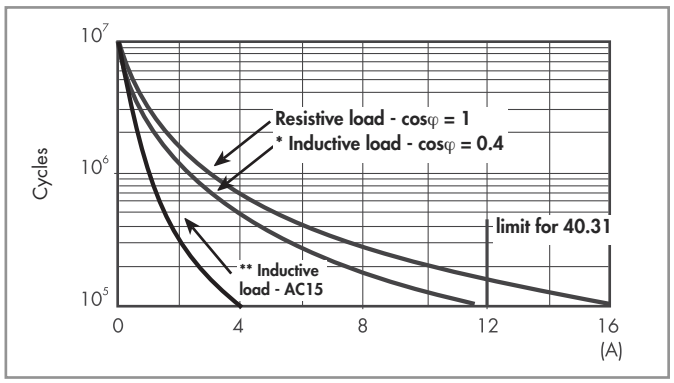
Terminal pin	Type	Coil version	A	B	C	D
PCB-relay, pin length 3.5 mm	40.31	DC sensitive	**1**	**0** - 3	**2**	**0** - 1
PCB-relay, pin length 3.5 mm	40.61	DC sensitive	1 - **2**	**0** - 3	**2**	**0** - 1
Plug in relay, pin length 5.3 mm	40.31	DC sensitive	**0**	**0** - 3	**0**	**0** - 1
Plug in relay, pin length 5.3 mm	40.61	DC sensitive	**0**	**0** - 3	**0**	**0** - 1

Technical data

Insulation according to EN 61810-1				
Nominal voltage of supply system		V AC	230/400	
Rated insulation voltage		V AC	250	400
Pollution degree			3	2
Insulation between coil and contact set				
Type of insulation			Reinforced (8 mm)	
Overvoltage category			III	
Rated impulse voltage		kV (1.2/50 µs)	6	
Dielectric strength		V AC	4,000	
Insulation between open contacts				
Type of disconnection			Micro-disconnection	
Dielectric strength		V AC/kV (1.2/50 µs)	1,000/1.5	
Conducted disturbance immunity				
Burst (5...50)ns, 5 kHz, on A1 - A2			EN 61000-4-4	level 4 (4 kV)
Surge (1.2/50 µs) on A1 - A2 (differential mode)			EN 61000-4-5	level 3 (2 kV)
Other data				
Bounce time: NO/NC		ms	2/5	
Vibration resistance (10...200)Hz: NO/NC		g	20/5	
Shock resistance NO/NC		g	20/5	
Power lost to the environment	without contact current	W	0.5	
	with rated current	W	1.2 (40.31)	1.8 (40.61)
Recommended distance between relays mounted on PCB		mm	≥ 5	

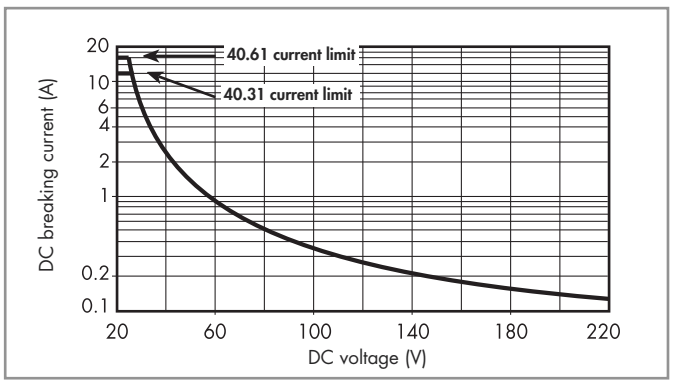
Contact specification

F 40 - Electrical life (AC) v contact current
Types 40.31/61



* Inductive load - cosφ = 0.4: inrush current = rated current
** Inductive load - AC15: inrush current = 10 x rated current

H 40 - Maximum DC1 breaking capacity



- When switching a resistive load (DC1) having voltage and current values under the curve, an electrical life of ≥ $100 \cdot 10^3$ can be expected.
- In the case of DC13 loads, the connection of a diode in parallel with the load will permit a similar electrical life as for a DC1 load.
  Note: the release time for the load will be increased.

Coil specifications

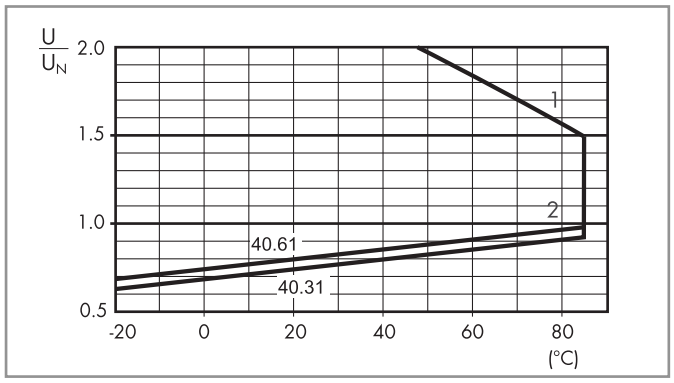
DC coil data - 0.5 W sensitive (type 40.31)

Nominal voltage U_N	Coil code	Operating range U_{min}	Operating range U_{max}	Resistance R	Rated coil consumption I at U_N
V		V	V	Ω	mA
12	**7**.012	8.8	18	288	42
24	**7**.024	17.5	36	1,150	21

DC coil data - 0.5 W sensitive (type 40.61)

Nominal voltage U_N	Coil code	Operating range U_{min}	Operating range U_{max}	Resistance R	Rated coil consumption I at U_N
V		V	V	Ω	mA
12	**7**.012	9.6	18	288	42
24	**7**.024	19.2	36	1,150	21

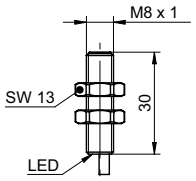
R 40 - DC coil operating range v ambient temperature



1 - Max. permitted coil voltage.
2 - Min. pick-up voltage with coil at ambient temperature.

Détecteurs de proximité inductifs IFRM 08P1701/L

Dessin d'encombrement



Données générales	
Type de montage	noyé
Portée nominale Sn	2 mm
Hystérésis de commutation	3 ... 20 % de Sr
Indication de l'état de sortie	LED rouge

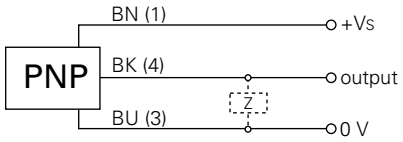
Données électriques	
Fréquence de commutation	< 5 kHz
Plage de tension +Vs	10 ... 30 VDC
Consommation max. (sans charge)	12 mA
Circuit de sortie	PNP à fermeture (NO)
Tension résiduelle Vd	< 2 VDC
Courant de sortie	< 200 mA
Protégé contre courts-circuits	oui
Protégé contre inversion polarité	oui

Données mécaniques	
Forme du boîtier	cylindrique avec filetage
Matériau (face active)	PBT
Matériau du boîtier	Acier chrome-nickel
Dimension	8 mm
Longueur du boîtier	30 mm
Version de raccordement	Câble, 2 m

Conditions ambiantes	
Température de fonctionnement	-25 ... +75 °C
Classe de protection	IP 67

Photo

Schéma de raccordement



μPMT Evaluation Kit
μPMT-EK2
TENTATIVE DATA SHEET

The μPMT-EK2 evaluation kit consists of a μPMT, high-voltage power supply and voltage divider circuit. The high-voltage power supply can be controlled via USB. The μPMT-EK2 allows for an easy way to evaluate the μPMT.

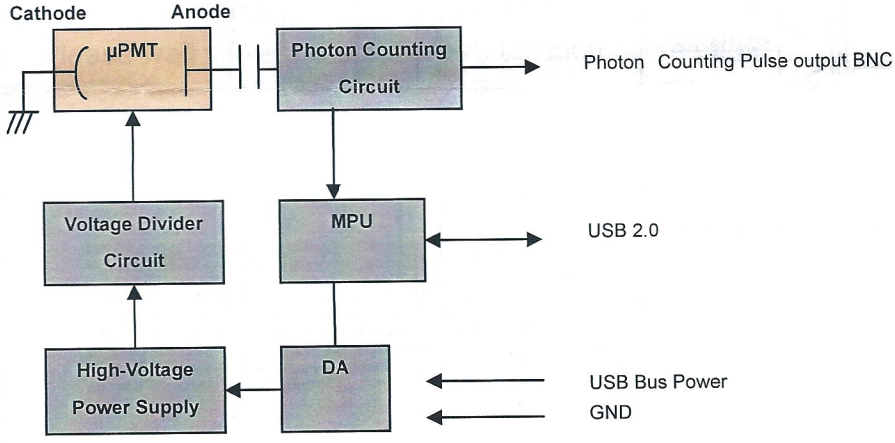
SPECIFICATIONS

Parameter		Value	Unit
Power Supply	Input Voltage	USB Bus Power	-
	Input Current	TBD	mA
PMT	Effective Area	3 x 1	mm
	Photocathode Material	Bialkali	-
	Spectral Response	280 ～ 600	nm
	Max. Average Anode Current	10	μA
Functions	Serial Interface	USB 2.0	-
	Compatible OS*1	Windows 7/ Windows 2000/ Windows XP Pro	-
Output	Pulse-Pair Resolution	Typ. 20	ns
	Pulse width	Typ. 10	ns
	Pulse Height(50 ohm Load)	Min. 2.0	V
Others	Operating Ambient Temperature *2	0 ～ + 50	°C
	Storage Temperature *2	- 20 ～ + 50	°C
	Weight	TBD	g

*1:Only supported with 32bit OS
*2:No condensation

Control software is supplied with the evaluation kit.

BLOCK DIAGRAM



HAMAMATSU
HAMAMATSU PHOTONICS K.K. Electron Tube Division
Jun.2013

Project BRAAVOO
Bioluminescence measure
Measurement sequence start at (GMT : +0) : Mon Jul 6 15:01:19 2015

Note : La mesure a commencé après 55 minutes dans l'obscurité

Time [s]	Hole n° 1	Time [s]	Hole n° 2	Time [s]	Hole n° 3	Time [s]	Hole n° 4	Time [s]	Hole n° 5	Time [s]	Hole n° 6	Time [s]	Hole n° 7	Time [s]	Hole n° 8	Time [s]	Hole n° 9	Time [s]	Hole n° 10
47	542	60	349	73	403	86	378	99	354	116	341	129	345	142	303	154	361	167	347
214	428	227	384	240	324	253	350	266	340	283	364	296	355	309	341	322	357	335	334
381	459	394	377	407	390	420	340	433	318	450	339	463	370	476	342	489	342	502	376
548	417	561	389	574	379	587	361	600	354	617	344	630	375	643	339	656	353	669	330
715	434	728	363	741	365	754	355	767	364	784	381	797	351	810	382	823	352	836	366
882	454	895	390	908	351	921	360	934	369	951	326	964	322	977	328	990	362	1003	340
1050	447	1063	391	1076	404	1089	336	1102	373	1118	379	1131	391	1144	385	1157	362	1170	351
1217	413	1230	403	1243	400	1256	380	1269	364	1286	388	1298	366	1311	356	1324	389	1337	378
1384	460	1397	337	1410	351	1423	365	1436	369	1453	409	1466	413	1479	400	1492	359	1505	389
1551	449	1564	389	1577	383	1590	362	1603	367	1620	406	1633	387	1646	358	1659	377	1672	378
1718	462	1731	439	1744	382	1757	377	1770	376	1787	436	1800	417	1813	392	1826	396	1839	392
1885	460	1898	447	1911	404	1924	390	1937	392	1954	405	1967	425	1980	393	1993	372	2006	413
2052	518	2065	417	2078	420	2091	405	2104	390	2121	397	2134	432	2147	381	2160	408	2173	386
2220	467	2233	435	2246	451	2259	412	2271	410	2288	433	2301	365	2314	450	2327	404	2340	396
2387	472	2400	451	2413	411	2426	399	2439	439	2455	429	2468	427	2481	415	2494	411	2507	404
2554	534	2567	454	2580	414	2593	453	2606	397	2623	430	2636	467	2648	398	2661	424	2674	395
2721	509	2734	438	2747	450	2760	445	2773	447	2790	445	2803	417	2816	426	2828	404	2841	420
2888	499	2901	464	2914	423	2927	460	2940	438	2957	453	2970	452	2983	387	2996	455	3009	429
3055	552	3068	475	3081	453	3094	407	3107	452	3124	454	3137	450	3150	447	3163	431	3176	416
3222	550	3235	456	3248	440	3261	464	3274	459	3291	466	3304	431	3317	475	3330	446	3343	444
3389	521	3402	498	3415	446	3428	464	3441	452	3458	413	3471	470	3484	440	3497	418	3510	468
3556	525	3569	461	3582	464	3595	424	3608	415	3625	467	3638	478	3651	464	3664	444	3677	447

Project BRAAVOO
Bioluminescence measure
Measurement sequence start at (GMT : +0) : Fri Jul 3 18:10:05 2015

Note : La mesure a commencé après 35 minutes dans l'obscurité

Time [s]	Hole n° 1	Time [s]	Hole n° 2	Time [s]	Hole n° 3	Time [s]	Hole n° 4	Time [s]	Hole n° 5	Time [s]	Hole n° 6	Time [s]	Hole n° 7	Time [s]	Hole n° 8	Time [s]	Hole n° 9	Time [s]	Hole n° 10
41	979	54	783	67	764	80	769	92	753	109	739	122	769	135	693	148	733	161	752
208	842	221	745	234	695	247	740	260	737	277	745	290	707	302	689	315	681	328	712
375	760	388	719	401	686	414	716	427	698	444	693	457	674	470	650	483	650	496	673
542	763	555	659	568	719	581	710	594	650	611	652	624	654	637	702	650	621	663	724
709	745	722	686	735	654	748	629	761	660	778	677	791	709	804	700	817	665	830	685
876	791	889	739	902	711	915	681	928	657	945	682	958	667	971	668	984	678	997	662
1043	682	1056	700	1069	658	1082	670	1095	629	1112	658	1125	678	1138	693	1151	653	1164	681
1211	733	1223	679	1236	693	1249	660	1262	637	1279	611	1292	632	1305	668	1318	620	1331	653
1378	717	1391	672	1404	686	1417	669	1429	699	1446	668	1459	631	1472	628	1485	660	1498	668
1545	712	1558	645	1571	664	1584	676	1597	666	1614	619	1626	639	1639	648	1652	679	1665	628
1712	704	1725	647	1738	672	1751	612	1764	634	1781	669	1794	655	1807	693	1820	622	1833	649
1879	718	1892	656	1905	670	1918	643	1931	639	1948	612	1961	652	1974	611	1987	641	2000	643
2046	713	2059	628	2072	650	2085	651	2098	593	2115	598	2128	646	2141	595	2154	607	2167	647
2214	680	2227	667	2240	633	2252	622	2265	621	2282	639	2295	591	2308	654	2321	661	2334	651
2381	727	2394	691	2407	655	2420	650	2433	622	2450	664	2463	647	2475	628	2488	572	2501	624
2548	681	2561	652	2574	639	2587	654	2600	641	2617	692	2630	707	2643	638	2656	665	2669	654
2715	708	2728	668	2741	641	2754	548	2767	619	2784	624	2797	653	2810	656	2823	614	2836	642
2882	693	2895	685	2908	641	2921	672	2934	661	2951	669	2964	668	2977	581	2990	643	3003	609
3049	674	3062	651	3075	621	3088	591	3101	644	3118	627	3131	673	3144	611	3157	598	3170	626
3217	688	3230	603	3242	645	3255	628	3268	607	3285	679	3298	598	3311	643	3324	657	3337	640
3384	662	3397	626	3410	647	3423	613	3435	653	3452	597	3465	590	3478	596	3491	626	3504	613
3551	679	3564	685	3577	608	3590	635	3603	638	3620	667	3633	639	3645	688	3658	594	3671	636
3718	698	3731	670	3744	650	3757	617	3770	651	3787	685	3800	594	3813	629	3826	604	3839	645
3885	672	3898	678	3911	611	3924	628	3937	621	3954	626	3967	642	3980	659	3993	618	4006	625
4052	727	4065	673	4078	639	4091	697	4104	611	4121	605	4134	628	4147	591	4160	616	4173	624
4219	673	4232	620	4245	592	4258	623	4271	627	4288	634	4301	672	4314	614	4327	623	4340	632
4386	704	4399	605	4412	622	4425	591	4438	605	4455	629	4468	610	4481	633	4494	603	4507	637
4554	688	4566	628	4579	615	4592	618	4605	600	4622	659	4635	593	4648	684	4661	599	4674	613
4721	696	4734	674	4747	669	4759	665	4772	657	4789	674	4802	654	4815	637	4828	617	4841	631
4888	708	4901	661	4914	631	4927	637	4940	625	4956	637	4969	585	4982	593	4995	674	5008	610
5055	699	5068	624	5081	641	5094	636	5107	623	5123	645	5136	676	5149	611	5162	648	5175	626
5222	716	5235	703	5248	636	5261	615	5274	601	5291	630	5304	636	5317	593	5330	638	5343	632
5389	641	5402	688	5415	627	5428	615	5441	621	5458	635	5471	653	5484	634	5497	617	5510	659
5556	740	5569	578	5582	636	5595	616	5608	602	5625	630	5638	637	5651	617	5664	650	5677	620
5723	694	5736	642	5749	637	5762	668	5775	601	5792	664	5805	654	5818	608	5831	621	5844	646
5890	660	5903	684	5916	635	5929	638	5942	618	5959	584	5972	646	5985	592	5998	633	6011	656
6057	651	6070	630	6083	631	6096	636	6109	619	6126	610	6139	646	6152	604	6165	621	6178	624
6225	676	6238	657	6251	628	6263	602	6276	588	6293	622	6306	608	6319	638	6332	667	6345	648
6392	687	6405	631	6418	620	6431	656	6444	627	6460	617	6473	642	6486	655	6499	636	6512	581
6559	722	6572	638	6585	636	6598	593	6611	641	6628	640	6641	627	6654	621	6666	590	6679	593
6726	720	6739	675	6752	617	6765	576	6778	611	6795	604	6808	617	6821	590	6834	623	6847	586
6893	692	6906	625	6919	616	6932	636	6945	614	6962	643	6975	641	6988	618	7001	613	7014	592
7060	700	7073	635	7086	619	7099	653	7112	627	7129	615	7142	613	7155	621	7168	609	7181	615
7227	694	7240	617	7253	607	7266	613	7279	625	7296	580	7309	606	7322	627	7335	576	7348	572
7394	700	7407	654	7420	578	7433	608	7446	638	7463	637	7476	624	7489	636	7502	576	7515	601
7562	678	7574	686	7587	631	7600	622	7613	623	7630	644	7643	593	7656	609	7669	632	7682	610

Project BRAAVOO
Bioluminescence measure
Measurement sequence start at (GMT : +0) : Fri Jul 3 14:25:01 2015

Note : La mesure a commencé après 15 minutes dans l'obscurité

Time [s]	Hole n° 1	Time [s]	Hole n° 2	Time [s]	Hole n° 3	Time [s]	Hole n° 4	Time [s]	Hole n° 5	Time [s]	Hole n° 6	Time [s]	Hole n° 7	Time [s]	Hole n° 8	Time [s]	Hole n° 9	Time [s]	Hole n° 10
41	1004	54	780	67	784	80	742	93	750	110	693	123	690	136	676	149	672	162	700
209	813	221	738	234	715	247	671	260	676	277	652	290	644	303	661	316	661	329	661
376	799	389	650	402	666	414	613	427	648	444	682	457	647	470	635	483	672	496	617
543	701	556	631	569	626	582	617	595	651	611	612	624	589	637	606	650	574	663	589
710	641	723	615	736	613	749	631	762	582	779	615	792	583	804	580	817	579	830	541
877	567	890	599	903	557	916	577	929	563	946	611	959	574	972	615	985	571	998	519
1044	629	1057	620	1070	573	1083	526	1096	543	1113	596	1126	569	1139	552	1152	564	1165	578
1211	611	1224	608	1237	572	1250	577	1263	561	1280	527	1293	565	1306	544	1319	543	1332	566
1378	572	1391	518	1404	581	1417	505	1430	522	1447	562	1460	560	1473	553	1486	560	1499	533
1545	602	1558	567	1571	544	1584	498	1597	550	1614	531	1627	563	1640	596	1653	526	1666	569
1713	581	1726	563	1738	554	1751	541	1764	546	1781	521	1794	488	1807	541	1820	527	1833	541
1880	639	1893	540	1906	560	1919	563	1932	574	1949	550	1961	524	1974	546	1987	485	2000	463
2047	617	2060	539	2073	587	2086	546	2099	521	2116	525	2129	537	2142	533	2155	542	2168	535
2214	597	2227	554	2240	547	2253	534	2266	484	2283	494	2296	525	2309	530	2322	523	2335	517
2381	587	2394	526	2407	537	2420	513	2433	490	2450	499	2463	518	2476	545	2489	545	2502	521
2548	665	2561	554	2574	546	2587	550	2600	513	2617	520	2630	521	2643	571	2656	492	2669	519
2716	617	2729	549	2741	531	2754	567	2767	518	2784	571	2797	530	2810	505	2823	528	2836	565
2883	594	2896	526	2909	531	2921	504	2934	525	2951	530	2964	549	2977	513	2990	535	3003	499
3050	563	3063	553	3076	532	3089	516	3102	501	3118	503	3131	500	3144	503	3157	529	3170	538
3217	570	3230	525	3243	520	3256	555	3269	537	3286	540	3299	527	3311	492	3324	475	3337	541
3384	610	3397	573	3410	479	3423	543	3436	524	3453	552	3466	526	3479	485	3492	512	3505	474
3551	580	3564	532	3577	486	3590	531	3603	544	3620	544	3633	513	3646	505	3659	509	3672	542
3718	595	3731	560	3744	559	3757	519	3770	520	3787	524	3800	549	3813	536	3826	506	3839	559
3885	594	3898	543	3911	568	3924	508	3937	529	3954	550	3967	528	3980	471	3993	516	4006	550
4053	581	4066	566	4079	515	4091	528	4104	496	4121	555	4134	466	4147	547	4160	522	4173	508
4220	598	4233	546	4246	533	4259	494	4272	563	4288	535	4301	534	4314	516	4327	501	4340	490
4387	608	4400	527	4413	509	4426	536	4439	557	4455	543	4468	549	4481	514	4494	519	4507	491
4554	570	4567	565	4580	523	4593	512	4606	548	4623	552	4636	543	4649	509	4661	504	4674	497
4721	597	4734	514	4747	481	4760	566	4773	526	4790	536	4803	537	4816	515	4829	534	4841	512
4888	598	4901	604	4914	567	4927	513	4940	503	4957	544	4970	541	4983	471	4996	541	5009	495
5055	604	5068	518	5081	508	5094	556	5107	484	5124	581	5137	516	5150	522	5163	488	5176	552
5222	648	5235	545	5248	565	5261	554	5274	510	5291	509	5304	565	5317	545	5330	547	5343	508
5389	601	5402	599	5415	536	5428	515	5441	535	5458	572	5471	541	5484	575	5497	513	5510	534
5556	658	5569	579	5582	485	5595	567	5608	522	5625	562	5638	537	5651	569	5664	507	5677	536
5723	654	5736	590	5749	550	5762	530	5775	517	5792	532	5805	500	5818	514	5831	523	5844	507
5890	591	5903	567	5916	533	5929	532	5942	556	5959	543	5972	520	5985	591	5998	521	6011	556
6058	614	6071	583	6083	560	6096	495	6109	515	6126	512	6139	568	6152	542	6165	479	6178	572
6225	618	6238	593	6251	545	6264	508	6276	530	6293	523	6306	570	6319	561	6332	498	6345	519
6392	628	6405	560	6418	575	6431	542	6444	517	6461	531	6473	549	6486	525	6499	508	6512	590
6559	606	6572	555	6585	507	6598	549	6611	548	6628	593	6641	559	6654	511	6667	566	6680	549
6726	642	6739	555	6752	557	6765	581	6778	559	6795	561	6808	556	6821	528	6834	506	6847	532
6893	626	6906	550	6919	548	6932	543	6945	557	6962	583	6975	549	6988	553	7001	549	7014	537
7060	616	7073	573	7086	546	7099	611	7112	552	7129	512	7142	607	7155	533	7168	549	7181	563
7227	622	7240	542	7253	596	7266	581	7279	583	7296	528	7309	527	7322	559	7335	548	7348	564
7394	624	7407	552	7420	563	7433	546	7446	525	7463	531	7476	528	7489	510	7502	573	7515	525
7561	623	7574	564	7587	520	7600	555	7613	501	7630	544	7643	565	7656	578	7669	570	7682	540

Project BRAAVOO
Bioluminescence measure
Measurement sequence start at (GMT : +0) : Thu Jul 2 12:00:49 2015

Note : La mesure a commencé après 1 minute dans l'obscurité

Time [s]	Hole n° 1	Time [s]	Hole n° 2	Time [s]	Hole n° 3	Time [s]	Hole n° 4	Time [s]	Hole n° 5	Time [s]	Hole n° 6	Time [s]	Hole n° 7	Time [s]	Hole n° 8	Time [s]	Hole n° 9	Time [s]	Hole n° 10
40	2525	53	2036	66	1802	79	1585	92	1496	109	1369	122	1293	135	1109	148	1105	161	1060
207	921	220	813	233	799	246	715	259	731	276	700	289	679	302	619	315	629	328	635
375	658	388	599	401	545	414	499	427	513	443	564	456	471	469	546	482	468	495	500
542	536	555	517	568	430	581	476	594	407	611	460	624	486	637	440	650	445	662	445
709	505	722	449	735	429	748	432	761	404	778	429	791	413	804	387	817	425	830	409
876	498	889	409	902	405	915	443	928	394	945	408	958	421	971	399	984	417	997	439
1043	519	1056	409	1069	430	1082	395	1095	419	1112	374	1125	414	1138	389	1151	393	1164	353
1211	461	1224	432	1236	399	1249	358	1262	363	1279	413	1292	424	1305	394	1318	377	1331	385
1378	472	1391	419	1404	393	1417	375	1430	398	1447	418	1460	373	1473	383	1485	403	1498	403
1545	501	1558	414	1571	383	1584	392	1597	368	1614	425	1627	387	1640	394	1653	388	1666	392
1712	460	1725	421	1738	370	1751	388	1764	414	1781	396	1794	388	1807	384	1820	404	1833	382
1879	496	1892	435	1905	413	1918	397	1931	386	1948	399	1961	390	1974	407	1987	368	2000	379
2046	493	2059	413	2072	401	2085	367	2098	411	2115	413	2128	398	2141	384	2154	390	2167	426
2214	475	2227	405	2240	417	2253	383	2265	389	2282	375	2295	421	2308	441	2321	394	2334	384
2381	502	2394	454	2407	371	2420	418	2433	398	2450	401	2463	403	2475	392	2488	380	2501	414
2548	483	2561	393	2574	372	2587	419	2600	431	2617	419	2630	453	2643	414	2656	412	2669	412
2715	485	2728	418	2741	426	2754	400	2767	425	2784	424	2797	405	2810	451	2823	412	2836	407
2882	496	2895	482	2908	394	2921	402	2934	411	2951	426	2964	398	2977	391	2990	401	3003	421
3049	464	3062	442	3075	404	3088	410	3101	430	3118	453	3131	424	3144	444	3157	429	3170	409
3217	455	3230	451	3242	445	3255	429	3268	464	3285	482	3298	437	3311	430	3324	422	3337	431
3384	496	3397	417	3410	446	3423	389	3436	419	3452	461	3465	458	3478	441	3491	433	3504	430
3551	536	3564	460	3577	457	3590	427	3603	480	3620	433	3633	457	3646	434	3658	424	3671	430
3718	459	3731	495	3744	446	3757	417	3770	453	3787	510	3800	434	3813	490	3826	467	3839	465
3885	514	3898	468	3911	475	3924	455	3937	452	3954	477	3967	462	3980	470	3993	431	4006	462
4052	539	4065	497	4078	489	4091	444	4104	447	4121	440	4134	455	4147	487	4160	491	4173	490
4219	535	4232	492	4245	469	4258	467	4271	454	4288	494	4301	485	4314	486	4327	472	4340	477
4387	576	4400	478	4413	492	4426	457	4438	460	4455	458	4468	428	4481	461	4494	463	4507	492
4554	557	4567	435	4580	543	4593	459	4606	458	4623	463	4635	492	4648	483	4661	457	4674	435
4721	580	4734	541	4747	511	4760	514	4773	489	4790	466	4803	471	4816	442	4829	479	4841	497
4888	535	4901	471	4914	489	4927	486	4940	450	4957	491	4970	499	4983	473	4996	458	5009	492
5055	552	5068	534	5081	493	5094	464	5107	464	5124	479	5137	493	5150	457	5163	505	5176	488
5222	545	5235	535	5248	458	5261	489	5274	504	5291	494	5304	471	5317	512	5330	489	5343	510
5389	570	5402	533	5415	499	5428	490	5441	515	5458	526	5471	473	5484	472	5497	497	5510	504
5557	622	5570	464	5582	509	5595	489	5608	511	5625	487	5638	504	5651	505	5664	470	5677	477
5724	555	5737	538	5750	494	5763	485	5776	495	5792	487	5805	510	5818	516	5831	472	5844	517
5891	558	5904	490	5917	506	5930	454	5943	494	5960	535	5973	509	5986	506	5998	503	6011	463
6058	617	6071	520	6084	483	6097	474	6110	451	6127	523	6140	514	6153	505	6166	488	6179	510
6225	514	6238	518	6251	568	6264	518	6277	508	6294	528	6307	537	6320	499	6333	481	6346	557
6392	580	6405	568	6418	517	6431	487	6444	503	6461	521	6474	518	6487	492	6500	510	6513	531
6559	601	6572	500	6585	529	6598	498	6611	530	6628	522	6641	523	6654	474	6667	540	6680	507
6726	598	6739	547	6752	521	6765	524	6778	544	6795	539	6808	553	6821	466	6834	505	6847	502
6893	665	6906	521	6919	513	6932	515	6945	517	6962	499	6975	497	6988	542	7001	517	7014	513
7061	634	7074	550	7087	562	7099	505	7112	511	7129	526	7142	525	7155	524	7168	535	7181	487
7228	635	7241	558	7254	570	7267	514	7280	512	7296	517	7309	544	7322	555	7335	508	7348	525
7395	608	7408	531	7421	496	7434	525	7447	532	7464	525	7477	516	7489	492	7502	539	7515	533
7562	594	7575	559	7588	534	7601	559	7614	511	7631	545	7644	491	7657	527	7670	527	7683	532

Explications
Nom de commande
Commande (1 à 8 Bytes)
Réponse (64 bytes, données en block de 4 bytes, LSB first)

Pour activer le module :
- setDefinedHV, setDefinedLLD, setIntegrationTime, setRepeat et StartCount

Pour désactiver le module :
- StopCount et setHV avec la valeur de 0

Commande	Valeur (ASCII)	Interprétation des réponses	Interprétation des réponses	Interprétation des réponses	Interprétation des réponses
getInfo					
50 4D 52 44	PMRD	***Byte [0-3]***	***Byte [4-7]***	***Byte [8-11]***	***Byte [12-15]***
50 4D 52 44 31 30 31 30 30 00 00 00 92 00 00 00		*Commande*	*?*	*?*	*SP (0x92 = 146 mA/W peak)*
61 00 00 00 E8 03 00 00 00 00 00 00 00 00 00 00		*SK (0x61 = 97 mA/W)*	*Plateau Voltage (0x03E8 = 1000V)*	*?*	*?*
30 00 00 00 08 00 00 00 33 30 31 36 30 30 39 32		*?*	*?*	*EK2 S/N*	*EK2 S/N*
30 41 30 34 37 30 37 34 32 30 31 34 30 34 32 36		*PMT S/N*	*PMT S/N*	*PMT S/N*	*PMT S/N*
getHV					
56	V				
56 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0x0 [V]	0 [V]		
getLLD					
4C	L				
4C 00 00 00 78 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0x178 [mV]	376 [mV]		
getIntegrationInterval					
49	I				
49 00 00 00 A0 86 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0x186A0 [10 µs]	100000 [10 µs]		
getRepetitions					
52	R				
52 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0x0	0		
setDefinedHV					
44 56	DV				
44 56 00 00 E8 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0x03E8 [V]	1000 [V]		
setDefinedLLD					
44 4C	DL				
44 4C 00 00 78 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0x178 [mV]	376 [mV]		
setIntegrationTime					
49 00 00 00 A0 86 01 00	I... †..	0x186A0 [10 µs]	100000 [10 µs]		
49 00 00 00 A0 86 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0x186A0 [10 µs]	100000 [10 µs]		
setRepeat					
52 00 00 00 00 00 00 00	R	0x0	0		
52 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0x0	0		
StartCount					
43	C				
00 00 00 00 1C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		data[0] = 0x1C	data[0] = 28		
01 00 00 00 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		data[1] = 0x16	data[0] = 22		
StopCount					
0D	CR				
0D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0x0	0		
setHV					
56 00 00 00 00 00 00 00	V	0x0 [V]	0 [V]		
56 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0x0 [V]	0 [V]		