L'analyseur nitrox façon "Open Source"

La mesure du pourcentage d'oxygène est incontournable de la plongée aux mélanges respirables enrichis en oxygène, ce que les plongeurs respirent sous le nom de nitrox. Cette mesure fait intervenir l'analyseur nitrox, un appareil qui convertit basiquement le signal issu d'un capteur chimique, la cellule oxygène, en l'affichage d'un pourcentage d'oxygène sur un écran. Le fait d'avoir affaire à un analyseur chimique en fait, en apparence, un produit technique compliqué avec suffisamment de voodoo derrière pour justifier le prix élevé généralement constaté sur cette gamme de produits (la plupart autour de 200-300 euros).

La réalité est beaucoup plus simple et ne nécessite aucun sacrifice humain si bien qu'il est possible de réaliser soi-même son propre analyseur pour une fraction du prix du commerce, sans nécessairement avoir des compétences solides en électronique. L'analyseur proposé ci-dessous peut techniquement rebuter un certain nombre de lecteurs puisqu'il est question de choses aussi abstraites que de la conversion analogique/digitale, de bus I2C et de composants programmables sur ordinateur. En réalité, tout est fourni et aucune connaissance particulière n'est requise.

En fait, l'investissement majeur du projet reste la cellule oxygène qu'on trouve autour de 100 euros. Sur ce projet , il s'agissait de recycler la cellule de référence Teledyne R17-VAN de mon analyseur précédent défunt après quelques années de bons et loyaux services. Le reste des pièces était plus ou moins de la récupération d'autres projets en ce qui me concerne et dans le cadre d'un montage à partir de rien, la liste des pièces ne dépasse pas la cinquantaine d'euros pour l'électronique pour un produit au final mieux que l'offre commerciale.

Un peu de théorie

Avant de s'atteler au bricolage, il est nécessaire de revenir sur le principe de fonctionnement d' un analyseur nitrox. Le fonctionnement électronique est extrêmement simple: la cellule oxygène délivre par un jeu de réactions chimiques une tension linéairement proportionnelle au pourcentage d'oxygène du mélange mesuré. La partie électronique de l'analyseur n'est en réalité qu'un voltmètre calibré pour lire cette tension et afficher non pas des millivolts mais le pourcentage d'oxygène. En connectant un multimètre aux bornes d'une sonde oxygène on a donc un analyseur à faible coup. Si le multimètre est suffisamment sensible car on parle en fait d'une résolution nécessaire au 1/100 de millivolts, que vous n'êtes pas ministre d'un gouvernement quelconque, il est parfaitement possible de calculer son pourcentage par le jeu d'une simple règle de 3.

Une cellule oxygène (Teledyne R17-VAN)

En pratique, c'est un peu plus compliqué puisque pour un gaz dont la concentration en oxygène est connue, la tension mesurée peut varier en fonction de différents facteurs (humidité, pression du flux de gaz sur la membrane). C'est là où les cours de physique du lycée et ma prof de chimie un peu chauve refont surface avec les fameuses "conditions standards de température et de pression" qu'en réalité on a jamais. C'est pour cela qu'il est nécessaire de procéder régulièrement à l'étalonnage de l'analyseur avant la mesure en le réglant sur un gaz de référence dont on connait la concentration en oxygène, la plupart du temps de l'air. L'air a l'avantage d'être abondant, facile à trouver et gratuit. Ce réglage revient en réalité à régler le facteur de conversion entre le voltage mesuré et le pourcentage d'oxygène affiché. La molette que l'on tourne sur les analyseurs pour dire qu'il y a 20,9% d'oxygène dans l'air fait ça. Tous les analyseurs sont construits sur ces principes.

Le circuit électronique

Le montage proposé ne fait pas exception au fonctionnement habituel du truc jaune et vert mais diffère des autres par le fait qu'il offre des fonctions supplémentaires sans effort: calibration automatique ou manuelle, affichage de la stabilisation de la mesure, un grand afficheur alphanumérique retro-éclairé et même sortie en continu de la mesure via USB ou pour quelques euros de plus via bluetooth.

Les analyseurs à fabriquer utilisent classiquement au moins un module voltmètre, des résistances à remplacer ou ajouter sur le module et un potentiomètre. Ma liste de course diffère grandement puisque j'utilise un module électronique programmable en langage C, qui va analyser des données issues d'un composant qui va convertir le voltage mesuré sur la cellule en données numériques.

Le module programmable est un circuit Arduino Uno. C'est une plateforme électronique de prototypage qui connaît un

énorme succès depuis le milieu des années 2000 chez les makers, ce mouvement qui allie bricolage classique à l'intégration de nouvelles technologies électroniques. Si vous avez aimé les Legos, que vous aimez l'électronique, vous adorerez les Arduinos si vous ne les connaissez pas déjà. Ces cartes électroniques disposent de connecteurs dans un format standardisé où de nombreuses broches peuvent-être programmées pour lire ou produire un signal électrique ou interpréter les informations de bus de données. Le positionnement des broches permet d'enficher des circuits électroniques spécialisées appelés Shields. L'afficheur de ce projet est un shield.

Arduino Uno

Le Shield LCD

L'Arduino et ses shields ont un énorme avantage: il y a peu ou pas de soudures soit un gain de temps énorme et le montage structurellement propre reste accessible aux personnes ayant très peu d'expérience en électronique. Pour utiliser l'Arduino et ses shields, il est nécessaire de charger un programme dans le circuit au travers d'un logiciel gratuit. Notre programme intègre à la fois la gestion de l'affichage, la mesure, la calibration, la detection de panne et la transmission USB ou bluetooth.

Une spécificité de cette analyseur est la calibration automatique. En théorie, n'importe quel modèle de sonde peut être utilisé ici ce qui en fait une autre différence majeure par rapport aux analyseurs commerciaux. Les sondes diffèrent entre elles de part leurs dimensions, leurs connecteurs, la résistance de charge et les gammes de tensions de sorties. Sur la cellule R17, le connecteur est une prise jack mono de 3.5mm et la résistance recommandée est de 100Kohm. Les essais sans cette dernière montrent que la mesure est possible également mais autant suivre les recommandations techniques du fabricant. A l'air libre, la cellule renvoie autour de 9 mV et de l'ordre de 45-50mV dans de l'oxygène pur. Dans ces conditions, on peut donc dire qu'une différence de 0.05mV correspond à une différence de 0.1% d'oxygène dans un gaz.

Ces ordres de grandeur sont petits donc il est nécessaire de disposer soit d'un circuit qui amplifie le signal électrique pour que des variations de 0.05mV soit mesurables, soit d'un circuit suffisamment sensible pour mesurer le signal. On va prendre une solution intermédiaire avec un circuit à la fois sensible et capable d'amplifier le signal. C'est en réalité un circuit convertisseur analogique vers numérique du petit nom de ADS1115. Ce circuit va mesurer la tension aux broches de la cellule oxygène dans une gamme comprise entre -0.256V et 0.256V et va renvoyer un nombre codé sur 16 bits sur un bus de données standardisé connu sous le nom de I2C, que l'Arduino décode sans problème. 1 bit représente donc 0.0078125mV. Ce pas est 6.4 fois plus petit que le pas de 0.05mV nécessaire pour représenter une variation de 0.1% d'oxygène ce qui permet une précision très largement suffisante.

L'ADS 1115

Le programme

Le noyau du programme de l'Arduino mesure la tension au borne de la sonde. La mesure est réalisée de très nombreuses fois par seconde donc on stabilise l'estimation du pourcentage via ce qu'on appelle une moyenne mobile. On prend les 10 dernières mesures et on calcule en permanence la moyenne de ces valeurs. Cela permet de limiter l'effet de toute sorte de mesures parasites. Le circuit fonctionne selon 3 modes:

1) Calibration automatique. Par défaut , à l'allumage, le gaz de référence est l'air. L'Arduino attend qu'il y ait 500 échantillons stables (le signal électrique varie de moins de 2% d'une mesure à l'autre) pour valider la calibration. Le processus prend une vingtaine de secondes. Le facteur de conversion mV vers O2 est simplement le rapport entre le pourcentage d'oxygène du gaz de référence et le nombre de millivolts mesurés pour ce même gaz.

2) Mesure en continue. Sur la base du facteur précédemment calculé, la mesure du pourcentage d'oxygène est affichée en permanence. Si la tension varie de moins de 2%, le symbole "====" apparaît ce qui indique une mesure quasi stable.

3) Calibration manuelle. Un écran permet de régler un autre gaz de référence avec les touches du clavier intégré au shield afficheur: les touches hautes/basses pour varier le pourcentage par pas de 1%, les touches gauche/droite par pas de 0.05% et la touche "select" pour valider et relancer une procédure de calibration.

La gestion de l'affichage est facilitée par l'existence d'une bibliothèque intégrée pour des afficheurs alphanumériques pilotés par un contrôleur HD44170. En parallèle quelques lignes de codes permettent d'envoyer les mesures sur le port USB et implicitement sur deux broches... qui sont connectées éventuellement à un petit circuit bluetooth "série". Cette connectivité ne servira pas pour la majorité des utilisateurs mais permet d'envisager d'autres applications nécessitant une mesure sans fil.

La liste des courses
Pièces du projet:

L'ensemble des pièces hormis la sonde nitrox peut être trouvée facilement sur ebay. Les mots clés à utiliser sont indiqués entre parenthèses

- Un arduino UNO ("Arduino Uno")

- Un convertisseur à base de ADS1115. ("ADS1115 breakout board"). Ne pas prendre le composant seul... qui ne fait que quelques mm² donc pas soudable pour un humain. Prendre obligatoirement une plaque sur lequel il est déjà soudé et qui comporte des broches notées SCA et SCL. Celui que j'utilise vient de chez TDVdesign.com (http://www.tdvdesign.com/ADS1115/). Un autre modèle facile à trouver est en vente chez Adafruit.com sous la référence 1085 (http://www.adafruit.com/products/1085). Les deux sont interchangeables.

- Un shield LCD alphanumérique avec clavier. Là aussi, différentes marques existent pour un même produit. ("LCD Keypad arduino shield")

- Un interrupteur à bascule (avec un capuchon étanche si le boitier est destiné à être étanche).

- Des fils électriques de couleurs différentes. On peut envisager comme je l'ai fait des fils avec connecteurs enfichables femelles dit "jumper wire" ou "cable jumper".

- Dans ce cas, il faut ajouter de la barrette sécable male au pas de 2.54 qu'on placera et soudera sur les différentes cartes. L'avantage de cette option est qu'en cas d'erreur, on a pas à dessouder puis ressouder des fils. 1 barrette suffit largement.

- Un connecteur pour la sonde. Sur la R17, c'est un prolongateur jack Male/Male et une prise jack femelle de 3.5mm.

- Une résistance optionnelle pour la sonde (100Kohm pour la R17).

- Un coupleur de pile et une pile.

- Un boitier (étanche ou non)

Outillage requis:

- Un petit tournevis

- Un fer à souder et l'étain pour la soudure

Assemblage
L'essentiel du travail est ici un travail de cablage selon le plan ci-dessous. Pour simplifier le travail il est recommandé de découper et souder les barrettes sécables sur le convertisseur et le shield. Il faut garder aussi à l'esprit que les soudures sont des points de contact potentiel: selon le boitier choisi il pourra être nécessaire de couvrir ces soudures avec de l'adhésif pour éviter un contact accidentel avec une surface métallique.

Le plan général du cablage (fait avec le logiciel gratuit Fritzing)

Le schéma général du circuit comporte les couleurs que j'ai utilisé.

- coupleur de pile: la broche rouge est connectée à l'une des broches de côté de l'interrupteur.

- orange/blanc (+ alimentation). Attention ce fil positif venant de la broche du milieu de l'interrupteur doit être marqué d'un signe distinctif pour ne pas être confondu avec le fil orange 5V puisqu'il délivre normalement 9V et peut détruire l'electronique s'il est connecté au mauvais endroit. Il est connecté au niveau de la broche VIN du shield afficheur LCD.

- orange (+5V): le fil d'alimentation du convertisseur. Il est relié entre la broche 5V du shield afficheur et la broche VDD du convertisseur

- bleu (+3.3V): le fil d'alimentation du circuit bluetooth optionnel. Il est relié entre la broche 3.3V du shield afficheur et la broche VDC du module bluetooth

- noir (- la masse). Le fil issu du coupleur de pile est connecté à la broches GND du module bluetooth et à l'une des deux broches à gauche de VIN sur le shield afficheur. La broche non utilisée est connectée à la broche GND du convertisseur. Ces fils alimentent les différents circuits de l'analyseur. Le convertisseur est cablé de la façon suivante:

- vert (bus I2C): le cable relie la broche SDA du convertisseur à la broche A4.

- jaune (bus I2C): le cable relie la broche SCL du convertisseur à la broche A5.

- rouge et blanc (sonde nitrox): faire une soudure de chaque cable à la prise de l'analyseur (ici jack 3.5mm male). Ne pas oublier de mettre une resistance de 100Kohm (pour une cellule Teledyne R17, voir fiche produit spécifique aux autres cellules) aux broches de la prise. Mesurer la tension avec un voltmètre réglé sur 50-200mV. Lorsque la tension est positive, brancher le cable relié à la borne + du voltmètre à la borne A0 et l'autre à A1. A l'exécution du programme, si la mesure est négative, inverser les broches.

Détail des connexions à réaliser sur le shield LCD

Le module bluetooth est quant à lui connecté selon le schéma principal. Sur un arduino, la broche RX correspond à la broche la plus à droite en haut sur le schéma. Cette broche est connecté à la broche TXD du module bluetooth. A l'inverse la broche TX (1) de l'Arduino est reliée à la broche RXD du module bluetooth. Il y a parfois une confusion entre TX et RX selon les fabriquants. Si aucun signal ne sort dès les premiers essais, il faudra probablement inverser les broches sur l'Arduino pour corriger le problème.

A partir de là, la partie électronique est complète. Prenez bien le temps de vérifier que les cables sont connectés aux bonnes broches. Le circuit n'est pas fonctionnel pour autant car il manque le programme qu'il faut charger dans l'Arduino.
L'installation du programme
Pour cela, il faut avant tout télécharger et installer le logiel Arduino IDE disponible en français sur cette page: https://code.google.com/p/arduino/downloads/list

N'importe quelle version fait l'affaire. A l'époque où j'ai rédigé cet article, la version choisie sous windows 7 est Arduino 1.0.5 R2. Comme pour tout logiciel, l'installation se fait en quelques clics. Je vous recommande d'accepter toutes les options proposées. Il est préférable de ne pas connecter l'Arduino avant d'avoir fini l'installation, celui-ci ayant besoin d'un driver qui va entre autre émuler le port série permettant la lecture ultérieure de l'analyse sur le port USB mais surtout permettre le chargement du programme dans la mémoire de l'Arduino.

Pour gagner du temps, avant de démarrer l'interface, il faut copier deux bibliothèques utiles à ce projet: Adafruit_ADS1X15 et RunningAverage. Les bibliothèques contiennent les fonctions nécessaires au bon fonctionnement de l'analyseur. Ces deux bibliothèques sont dans le fichier compressé suivant: lib-nitrox.zip .

Il faut ouvrir ce fichier (Winzip, Winrar, etc). Il contient un répertoire par bibliothèque. Dans l'explorateur de fichier, repérer sur l'ordinateur où est le répertoire Arduino. Sous Windows 7 64 bits, il est probablement dans C:\Program Files (x86)\Arduino

Dans le répertoire Arduino, allez dans le sous répertoire "libraries" (dans mon cas C:\Program Files (x86)\Arduino\libraries) et y déposer les répertoire des bibliothèques.

On connecte ensuite l'Arduino. L'ordinateur devrait le détecter automatiquement et au bout de quelques instants, dans le gestionnaire de périphérique, un nouveau port série USB doit être visible.

On démarre ensuite l'interface Arduino. Si aucun raccourci n'a été crée, le programme est dans le répertoire Arduino (C:\Program Files (x86)\Arduino) sous le nom Arduino.exe sous Windows.

Faites un copier-coller du code ci-dessous dans l'éditeur:

```
////////////////////////////////////////////////////////////////////////////////

// Arduitrox version 1.0

// Main code by Lionel Pawlowski - Copyright (c) 2014

// ADS library by Adafruit

// Running average library by Rob Tillaart

// Commercial use and/or distribution of this code and relevant

// hardware forbidden without prior agreement.

// Provided "as is". Use at your own risk
```

```
////////////////////////////////////////////////////////////////////////////

#include <Wire.h>
#include <LiquidCrystal.h>
#include <Adafruit_ADS1015.h>
#include <RunningAverage.h>



Adafruit_ADS1115 ads;
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

int opmode=0;
int cnt;
float oldRA;
float sig1,sig,sig2;
float relpourc;
float gain = 0.0078125;
float pour;
RunningAverage RA(10);
int samples=0;

// Définition du gaz de référence
float calibgas=20.95;

void setup(void)
{
// Message d'accueil si besoin (nom, etc)
lcd.begin(16, 2);
lcd.setCursor(0,0);
lcd.print("Arduitrox v1.0");
lcd.setCursor(0,1);
lcd.print("");
```

```
delay(3000);
Serial.begin(9600);
ads.setGain(GAIN_SIXTEEN);    // 16x gain   1 bit = 0.0078125mV
ads.begin();
oldRA=0;
cnt=0;
lcd.setCursor(0,0);
lcd.print("Cal. auto");
lcd.setCursor(0,1);
lcd.print("               ");
}



void loop(void)
{
int16_t adc0;
oldRA=RA.getAverage();
adc0 = ads.readADC_Differential_0_1();
RA.addValue(adc0);
cnt=cnt+1;
sig=(RA.getAverage()-oldRA)/oldRA;
sig1=abs(sig);

// Calibration Auto
if (opmode==0) {
if (sig1<0.0002) {   // 0.0002
          samples=samples+1;
} else {
//  samples = 0;
}
lcd.setCursor(0,1);
lcd.print(RA.getAverage()*gain,2);
```

```
lcd.print("mV ");
lcd.print(calibgas,2);
lcd.print("% ");
lcd.setCursor(10,0);
pour=samples/5;
lcd.print(pour,0);
lcd.print("%  ");
if ((RA.getAverage()*gain)<0.02) {
        lcd.setCursor(0,0);
lcd.print("PANNE CELLULE O2");
opmode=10;
}
Serial.print("CAL,");
Serial.print(cnt);
Serial.print(",");
Serial.print(RA.getAverage()*gain,4);
Serial.print(",");
Serial.println(calibgas);
if (samples==500) {
samples=0;
opmode=2;
lcd.setCursor(0,0);
lcd.print("CALIBR TERMINEE");
lcd.setCursor(0,0);
delay(3000);
relpourc=100*calibgas/RA.getAverage();
lcd.setCursor(0,0);
lcd.print("              ");
lcd.print("              ");
lcd.begin(16, 2);
delay(500);
}
}
```

```
// Lecture des échantillons
if (opmode==2) {
lcd.setCursor(0,0);
lcd.print("%O2: ");
pour = RA.getAverage()*relpourc/100;
lcd.print(pour,1);
lcd.print(" ");
lcd.setCursor(0,1);
lcd.print("mV : ");
lcd.print(RA.getAverage()*gain,2);
lcd.print(" ");
lcd.setCursor(10,0);
if (sig1<0.0002) {
        lcd.print("====");
} else {
lcd.print("    ");
}
Serial.print("MES,");
Serial.print(cnt);
Serial.print(",");
Serial.print(RA.getAverage()*gain,4);
Serial.print(",");
Serial.println(pour,2);
delay(200);
}

// Calibration manuelle
if ((analogRead(0)>600)&&(analogRead(0)<700)) {
    if (opmode<3) {
    opmode=3;
samples=0;
```

```
lcd.setCursor(0,0);
lcd.print("Calibr. manuelle");
lcd.setCursor(0,1);
lcd.print("                ");
delay(500);
} else {
lcd.setCursor(0,0);
lcd.print("                ");
lcd.setCursor(0,1);
lcd.print("                ");
delay(2000);
lcd.setCursor(0,0);
lcd.print("Cal. auto");
lcd.setCursor(0,1);
lcd.print("                ");
opmode=0;
}
}
if (opmode==3) {
lcd.setCursor(0,1);
lcd.print("%O2 REF: ");
lcd.setCursor(9,1);
lcd.print(calibgas,2);
lcd.print("  ");
if ((analogRead(0)==100)&&(calibgas<99)) {
      calibgas=calibgas+1;
delay(100);
}

if ((analogRead(0)==0)&&(calibgas<100)) {
      calibgas=calibgas+0.05;
delay(200);
}
```

```
if ((analogRead(0)==257)&&(calibgas>1)) {
calibgas=calibgas-1;
delay(100);
}
if ((analogRead(0)==411)&&(calibgas>0.05)) {
calibgas=calibgas-0.05;
delay(200);
}
}
}
```

Une fois le programme copier, il faut vérifier le code en cliquant sur le bouton de vérification du code.

Puis on le "téléverse". Cette traduction bizarre signifie qu'on charge le programme dans l'Arduino. Au bout de quelques seconds, le chargement est fini normalement et il ne doit pas y avoir de message d'erreur (autrement le texte de la fenêtre noir devient orange au lieu de blanc)
Si le chargement ne se fait pas, vérifier que l'Arduino est bien visible. Sur la capture d'écran ci-dessus, il est visible sur le port COM13 et c'est effectivement un Arduino Uno sur lequel je travaille.

Vous pouvez modifier le type de carte et le port en allant dans le menu "Outils".

Une fois le chargement terminé. Votre projet est autonome. En allant sur "Outils", moniteur série, l'analyseur va démarrer et afficher les valeurs mesurées. En déconnectant le circuit et en allumant normalement, vous devez voir les informations s'afficher sur l'afficheur alphanumérique. Dans le cas contraire, le contraste de l'afficheur est peut-être trop faible. Vous pouvez regler celui-ci avec un petit tournevis plat en jouant sur la vis de réglage en haut à gauche de l'afficheur.

Sinon si tout va bien, vous devriez voir un affichage de ce type:

Le projet une fois complété et opérationnel

Conclusion et perspectives...
Le code "open source", le nombre de broches restantes font que ce projet peut évoluer vers d'autres choses. Par exemple, il reste des broches dispo pour la mesure trimix ou la mesure de la pression. Les sondes trimix ou pour d'autres gaz (CO, CO2...) reposent sur des principes différents et nécessitent un peu plus d'électronique. Les sondes pour hautes pressions délivrent pour la plupart des signaux analogiques qu'on peut mesurer avec un autre convertisseur. Il faut noter que le bus de données I2C permet aussi de connecter d'autres circuits convertisseur. Bref, beaucoup de possibilités d'expérimentation et d'évolution.