Gaussmètre numérique

Institut Gériatrique des Solex
Bernique

Abstract

Gaussmètre numérique permettant de mesurer le champ magnétique des volants de cyclomoteurs (à Galet bien sûr!) après auto-calibration.

Keywords: Arduino pro mini, UGN3503, écran OLED

Table des matières

I. Introduction

Il arrive parfois que l'apprenti mécano en devenir se pose des questions métaphysiques quant à l'état de l'aimantation du volant magnétique de son Solex, surtout si celui-ci est un modèle sans embrayage connu par ailleurs pour perdre son aimantation avec le temps… source de panne potentielle côté allumage.

Ce document décrit la construction d'un système permettant de mesurer le champ magnétique proche des pôles présents dans le volant magnétique. Au delà de la mesure que je ne saurais qualifier d'absolue, l'intérêt réside surtout dans la comparaison des lectures d'un volant à l'autre afin de se rassurer.

II. Composants

1 Le cerveau du système

Bien connu des bidouilleurs de tous poils, la plateforme Arduino est le support idéal pour réaliser la tête pensante de ce projet : open source, disponible partout (y compris dans des déclinaisons chinoises pour quelques piastres), bénéficiant d'une immense communauté pour le support, d'un langage propre et de bibliothèques fournies pour ceux qui ne souhaitent pas s'aventurer dans la pureté d'une programmation en C des registres de son µ-processeur AVR de chez Atmel®, il est d'un accès très aisé.

J'ai choisi ici sa déclinaison en modèle Arduino Pro Mini pour ses dimensions restreintes et un nombre d'entrées/sorties largement suffisants (limite indécent!). Tant qu'à y être, choisissez un modèle à oscillateur (ci-contre) plutôt qu'à quartz, c'est encore moins cher et d'une précision largement suffisante pour notre application!

On pourrait imaginer d'utiliser également une carte à base de ATmega168 encore moins chère… mais il faudra alors s'assurer de la compatibilité du code donné ici (dont la bibliothèque).

Le code à charger en mémoire est donné en annexe (les conditions de transfert sont données en en-tête du programme).

2 Alimentation

Vu qu'il s'agit d'un appareil d'atelier à vocation portative, on va opter pour une bonne vieille pile 9V qui fournira le jus en quantité suffisante sur l'entrée RAW (ou Vin) de l'Arduino. Le convertisseur présent sur la carte nous donnera en retour le 5v sur la pin Vcc (ou 5v), nécessaire pour alimenter le capteur et l'écran.

3 Capteur

Dans la série des capteurs à effet hall linéaires, il y a pléthore… j'ai jeté mon dévolu sur le UGN3503UA qui présente le bon goût d'être disponible sur eBay pour quelques centimes malgré son obsolescence.

Comme il ne s'agit pas de faire un capteur absolu dernier cri, c'est amplement suffisant.

Le signal sera envoyé sur la pin A0 (analogique).

4 Écran OLED

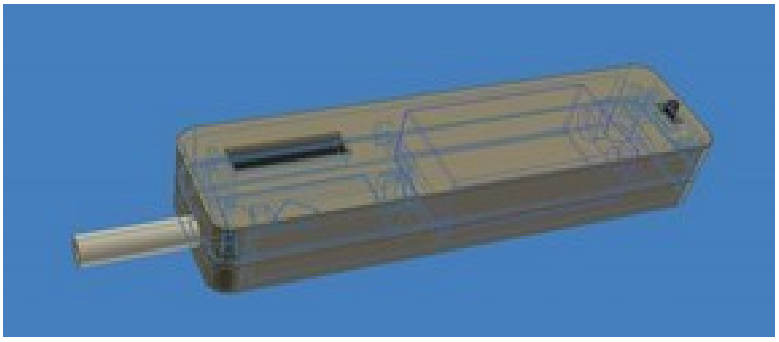
Histoire de faire un appareil qui tient dans la main (assez pratique pour mesurer un peu partout dans des endroit pas toujours accessibles), j'ai jeté mon dévolu sur un petit écran OLED 128x32 qui traînait dans une boite (moins de 2€ sur eBay).

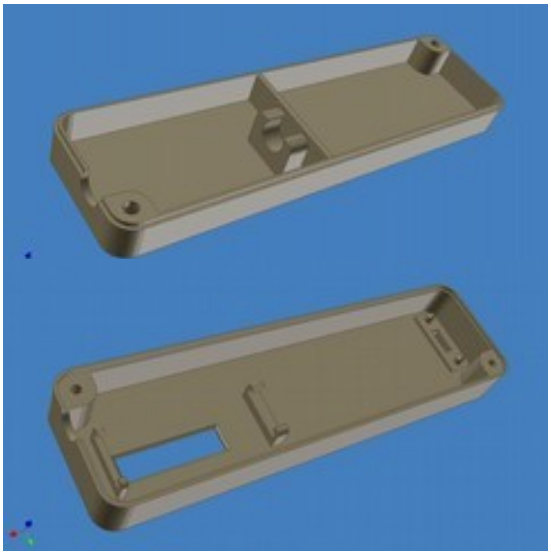
Sa programmation est un peu plus complexe qu'un traditionnel écran LCD malgré le protocole I2C supporté et les librairies disponibles sur le web, mais les possibilités d'affichage et le rendu valent le coup ! Il monopolisera les pin A4 et A5 de l'Arduino.

5 Boîtier

Pendant qu'on y est, tant qu'à faire un prototype, on réalisera le boîtier en impression 3D afin de minimiser encore une fois le volume total du boîtier et s'adapter au mieux aux différents composants.

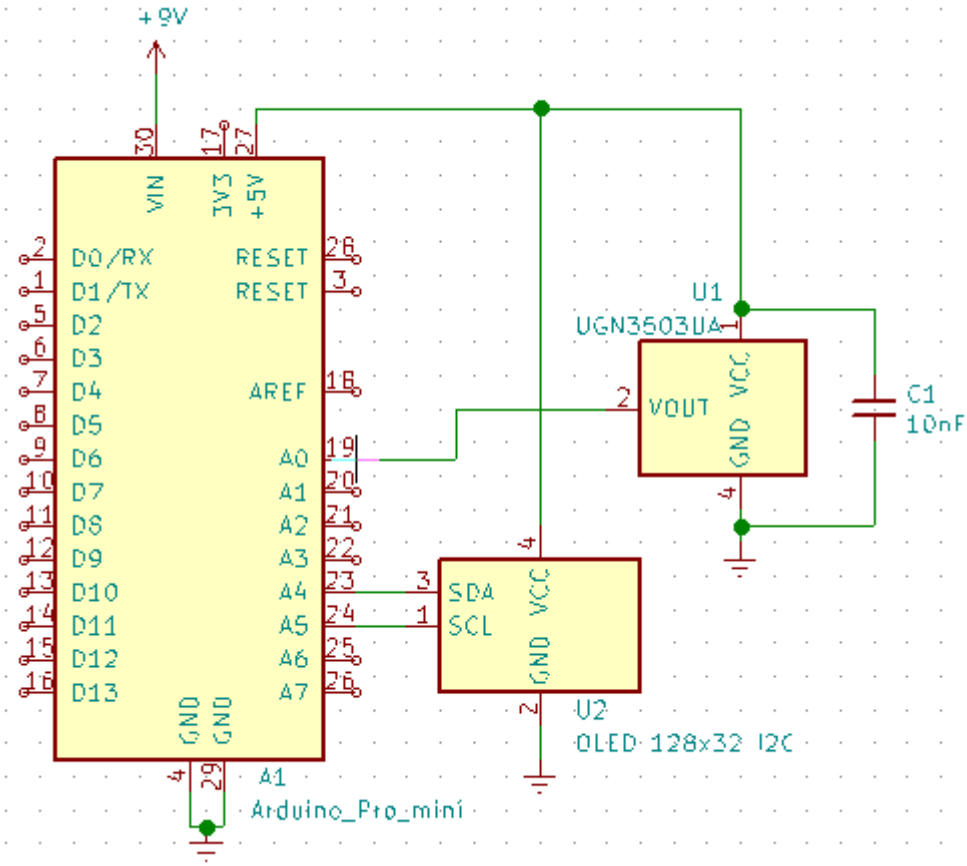
Après, rien ne vous empêche de squatter un boîtier déjà existant…





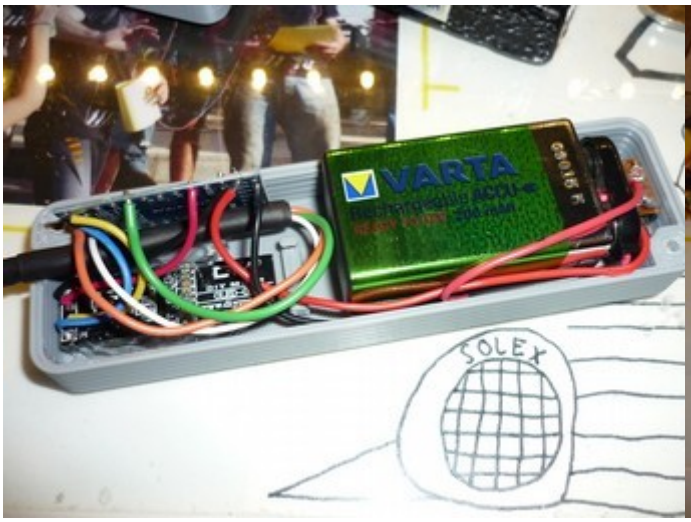
III. Le circuit

On va faire simple : en plus des éléments précités, il vous faut juste un condensateur céramique de 10nF qui permet de stabiliser un peu plus la mesure et un interrupteur général de mise sous tension (non représenté sur le schéma ci-dessous).



IV. Mise en œuvre

Rien de bien sorcier… suffit de faire chauffer la station de soudage et de programmer le µ-processeur via l'interface Arduino (codage en langage Arduino afin de bénéficier de la bibliothèque pour l'écran OLED… désolé pour les puristes du C !).



V. Conclusion

Et woulà, ça fonctionne…

Annexe : le code

```
/*******************************************************************************
*  Gaussmètre                                                                  *
*  Mesure champ magnétique (aimants, volant magnétique…)                       *
*  Affichage sur écran OLED 128x32 monochrome blanc, I2C                       *
*                                                                              *
*  Processor: Arduino pro mini – 16MHz                                         *
*  Compiler Arduino ISP                                                        *
*  Programmer: FTD1232                                                         *
*                                                                              *
*                        _____________________                                 *
*                       |   *   *   *   *   *   *  |                           *
*              PD1-1    *|                         |*       RAW      <- 9VDC   *
*              PD0-0    *|                         |*       GND                *
*              PC6-RESET*|                         |*       PC6-RESET          *
*              GND      *|        PC4-18 (A4) *    |*       VCC                *
*              PD2-2 INT0*|       -> SDA           |*       PC3-17  (A3)       *
*              PD3-3 INT1*|                        |*       PC2-16  (A2)       *
*              PD4-4    *|        PC5-19 (A5) *    |*       PC1-15  (A1)       *
*              PD5-5    *|        -> SCL           |*       PC0-14  (A0) <- signal*
*              PD6-6    *|                         |*       PB5-13             *
*              PD7-7    *|                         |*       PB4-12             *
*              PB0-8    *|                         |*       PB3-11             *
*              PB1-9    *| _______________________ |*       PB2-10             *
*                                                                              *
*                                                                              *
* Bernique                                                                     *
* Common Creative BY-NC-SA                                                     *
* Jan.2019                                                                     *
*                                                                              *
* Status: OK                                                                   *
*******************************************************************************/

#include <Arduino.h>
#include <U8g2lib.h>

#ifdef U8X8_HAVE_HW_SPI
#include <SPI.h>
#endif
#ifdef U8X8_HAVE_HW_I2C
#include <Wire.h>
#endif

int i=0;                        // compteur
int champ=0;            // mesure analogique capteur 0-5v (0-1024 points)
int wait=250;           // temps en ms entre deux mesures
long calibrate = 0;             // valeur de calibration du capteur au démarrage versus environnent
float resolution = 3.756;// résolution du capteur en gauss/point (1024 points entre 0 et 5v)
int calib_lenght=256;           // quantité de mesures utilisées pour effectuer la calibration

// définition écran OLED SSD1306 128x32 monochrome blanc
U8G2_SSD1306_128X32_UNIVISION_F_HW_I2C u8g2(U8G2_R0, /* reset=*/ U8X8_PIN_NONE, /* clock=*/ SCL, /* data=*/ SDA);
```

```
void setup(void) {
 u8g2.begin();
 u8g2.enableUTF8Print();                  // enable UTF8 support for the Arduino print() function

 u8g2.setFont(u8g2_font_t0_12b_tr);       // define font type https://github.com/olikraus/u8g2/wiki/fntlistall
 u8g2.setFontDirection(0);
 u8g2.clearBuffer();

 // Affichage écran acceuil #1
 u8g2.setCursor(0, 15);
 u8g2.print("Gaussmetre v1.0");
 u8g2.setCursor(0, 27);
 u8g2.print(" by Bernique");
 u8g2.sendBuffer();
 delay(1750);

 // Calibration capteur et affichage écran #2
 calib();
}

void loop(void) {

 while (1){

  // mesure et affichage résultat écran #3
  mesure();
  // gestion affichage disque tournant 1/2
  u8g2.setDrawColor(1);
  u8g2.drawDisc(118, 15, 6, U8G2_DRAW_UPPER_RIGHT);
  u8g2.drawDisc(118, 15, 6, U8G2_DRAW_LOWER_LEFT);
  u8g2.setDrawColor(0);
  u8g2.drawDisc(118, 15, 6, U8G2_DRAW_LOWER_RIGHT);
  u8g2.drawDisc(118, 15, 6, U8G2_DRAW_UPPER_LEFT);
  u8g2.sendBuffer();
  delay(wait);

  // mesure et affichage résultat écran #3
  mesure();
  // gestion afichage disque tournant 2/2
  u8g2.setDrawColor(1);
  u8g2.drawDisc(118, 15, 6, U8G2_DRAW_LOWER_RIGHT);
  u8g2.drawDisc(118, 15, 6, U8G2_DRAW_UPPER_LEFT);
  u8g2.setDrawColor(0);
  u8g2.drawDisc(118, 15, 6, U8G2_DRAW_UPPER_RIGHT);
  u8g2.drawDisc(118, 15, 6, U8G2_DRAW_LOWER_LEFT);
  u8g2.sendBuffer();
  delay(wait);
 }
}

void calib() {
 u8g2.clearBuffer();
 u8g2.setCursor(0, 15);
 u8g2.print("Calibration capteur");
 u8g2.drawFrame(0,20,128,7);
 u8g2.sendBuffer();

 // calibration par moyennage de mesures
```

```
    while (i<calib_lenght) {
      u8g2.drawBox(0,20,ceil(i/(calib_lenght/128)),7);        // barre de défilement
      u8g2.sendBuffer();

   calibrate=calibrate + analogRead(A0);
   i++;
   }
  calibrate = ceil(calibrate/calib_lenght);                  // moyenne
}

void mesure() {
 champ = analogRead(A0);

 u8g2.setDrawColor(1);
 u8g2.clearBuffer();

 u8g2.setCursor(0, 15);
 u8g2.print("Champ: ");
 u8g2.print(abs((champ-calibrate)*resolution),0);
 u8g2.print(" gauss");

 u8g2.setCursor(0, 27);
 u8g2.print("Pole: ");
 if (champ>calibrate+1)  u8g2.print("SUD ");
 if (champ<calibrate)  u8g2.print("NORD");

 u8g2.drawCircle(118, 15, 9, U8G2_DRAW_ALL);
 u8g2.sendBuffer();
}
```