Le Milliohmmètre 0 - 10Ω

Besoins

Les appareils de mesure facilement accessibles à Monsieur Tout-le-monde, numériques ou surtout analogiques, sont incapables de lire correctement les faibles valeurs de résistances, inférieures à 10 Ω, les valeurs de l'ordre de l'ohm sont souvent totalement fantaisistes. Les valeurs rencontrées en électricité moto sont souvent inférieures à 15 Ω, j'ai donc entrepris la réalisation d'un milliohmmètre.

Pour ce faire j'utilise une carte « mini-voltmeter » de chez Tuxgraphics dont je récupère une partie du logiciel (la gestion de l'affichage, tordue, en raison d'une curieuse distribution des ports).

C'est une carte à microcontrôleur Atmel AT Mega8 qui m'intéresse en raison de :

- puce à convertisseur A/D intégré de 10 bits (1024 incréments de mesure)
- 3 digits d'affichage
- se programme sous Linux (gratuit)
- on compile le c et on charge la puce avec des outils gratuits sous Linux (en fait le câble de programmation peut se réaliser, sur port parallèle, avec deux résistances ! , c'est de plus totalement gratuit, m'enfin, c'est que des kopeks pour les résistances)
- circuit imprimé et composants disponibles chez Tuxgraphics (moins de 12 € en juillet 2012, y compris les afficheurs LED et le microprocesseur)

Les trois digits d'affichage permettent d'afficher les centi-ohms ou les 10/1000 d'ohm. Ce qui me paraît suffisant et dispense de devoir régler le zéro.

Spécifications

Un ohmmètre classique se compose d'un générateur de courant constant qui débite dans la résistance à mesurer. Selon ce bon Georg (Ohm), la tension apparaissant aux bornes d'une résistance dépend du courant par la célèbre loi **U=RxI**.

Pour pouvoir utiliser les 10 bits au mieux, il faut que l'amplitude de la tension à mesurer corresponde à la tension de référence de mesure. Le mega8 peut utiliser une référence interne à 2,56V, ce que j'ai choisi, car utiliser la tension d'alimentation 5V, même régulée, est moins précis. Pour avoir une lecture directe il faut donc un courant de 250mA, qui est relativement important, mais l'utilisation en temps partiel conduit à une faible dissipation moyenne dans la résistance à mesurer. Il faudra tenir compte de l'échauffement dans la résistance à mesurer pour les résistances très fragiles, ou pour la perte éventuelle de précision. Eviter l'amplificateur extérieur. Tout se passe (mesure et calcul) hors phases d'affichage.

Le générateur de courant

Le générateur de courant est fait par un régulateur de tension ajustable (LM337) qui débite dans une résistance. La valeur de la résistance est calculée par la loi de Georg en prenant en compte la référence interne de 1,25V, il faut ajouter un petit élément variable pour compenser le courant consommé par l'électrode de commande.

Il n'est pas question d'avoir un paramètre multiplicateur d'ajustement logiciel sous peine d'avoir un incrément d'affichage supérieur à 1 par pas de mesure. Pour laisser le générateur de courant travailler correctement (il reste 2,4V entre l'entrée de mesure et l'alimentation 5V lorsqu'on est à Vmax, insuffisant) le générateur de courant est alimenté par l'alimentation primaire, ici du +12V de batterie ou provenant d'un transfo. La résistance sera de 5Ω , ajustable, Une diode Zener limite la tension de sortie du générateur à 5,15V pour ne pas endommager l'entrée du convertisseur A/D, en réalité c'est ici une super-Zener faite avec une Zener et un transistor de puissance.

Le générateur de courant de 0,25A est commandé par le processeur, il génère une puissance de 2,56W à conduction de 100 %, Ici, il n'est généré que pendant 3,6 % du temps, donc une puissance dans la résistance de 9mW moyens. Pas de problèmes attendus de ce côté dans le domaine de l'électricité moto.

Dans le cas de mesure de circuits fortement inductifs (dynamos par exemple), la mise à la masse de PC5 par un bouton temporaire augmente le temps d'établissement qui passe à 50ms avant la mesure effective. Le courant est alors généré pendant 50ms et pourrait échauffer certains éléments sensibles,

L'affichage

Comme on ne dispose que de 3 digits d'affichage, pourquoi ne pas utiliser le MSD (Most Significant Digit, l'afficheur le plus à gauche) en mode hexadécimal ? Un afficheur classique 3 digits affiche jusqu'à 999, celui-ci affiche jusqu'à 1599, donc au lieu de mesurer jusqu'à 9,99 Ω, on pourrait aller à 15,99 Ω.

OK il faut lire l'hexadécimal, mais ce n'est que sur un digit, en voici l'alphabet :

999	9,99 V
A99	10,99 V
B99	11,99 V
C99	12,99 V
D99	13,99 V
E99	14,99 V
F99	15,99 V

Ceci pour le principe, mais dans le cas présent, on est limité par la résolution du convertisseur A/D sur 1024 états, donc la valeur max sera 1023 (ne pas oublier que 0 est un état significatif). Cette astuce est donc un conseil pour dépasser de 60 % un affichage limité pour d'autres montages ; ici, elle permet de dépasser légèrement la valeur de 10 ohms et de ne pas la confondre avec la valeur max.

Voici donc la valeur max lue en milliohmmètre : **A23** (= 1023) et correspond à la tension de référence interne soit 2,56V

Le programme

Ecrit en c, compilé par avr-gcc.

Phases de fonctionnement :

- validation du générateur de courant
- attente 2ms pour stabilisation, et pour charger le condensateur antiparasite
- attente 50ms lorsque le bouton est appuyé, pour stabilisation sur charges fortement inductives (inducteur dynamo)
- mesure de la tension par le convertisseur A/D
- inhibition du générateur de courant

La tâche de fond consiste à rafraîchir l'affichage LED

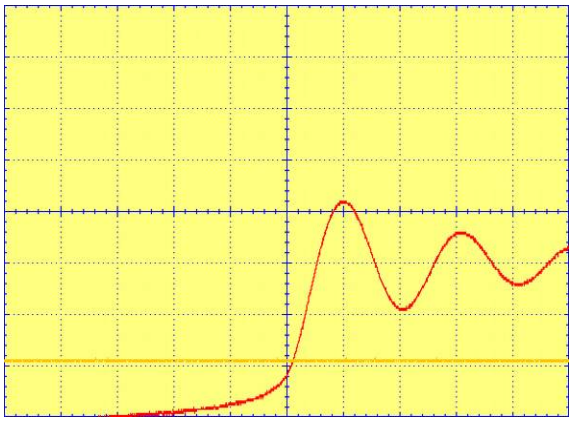
Un écrêteur logique limite l'affichage à 1023 (A23 ici), c'est l'affichage sans résistance connectée.

Autre usage

L'impulsion de validation du générateur de courant est utilisée, via une diode et un condensateur étalonné, pour mesurer facilement (avec un oscilloscope) une inductance par fréquence de résonance. Par exemple, et au hasard, une bobine d'allumage.

Bobine Lucas 17M12 avec adaptateur

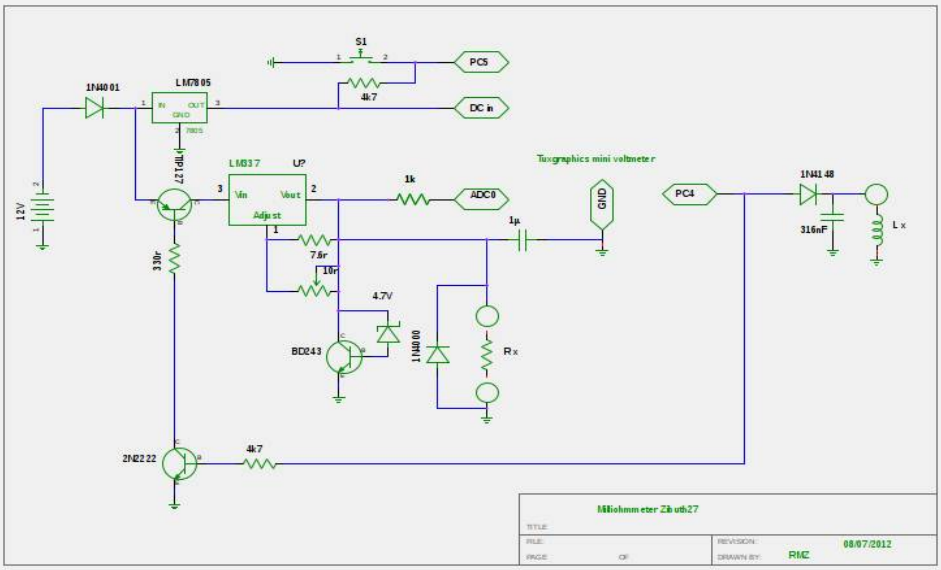
x=2ms y=1V

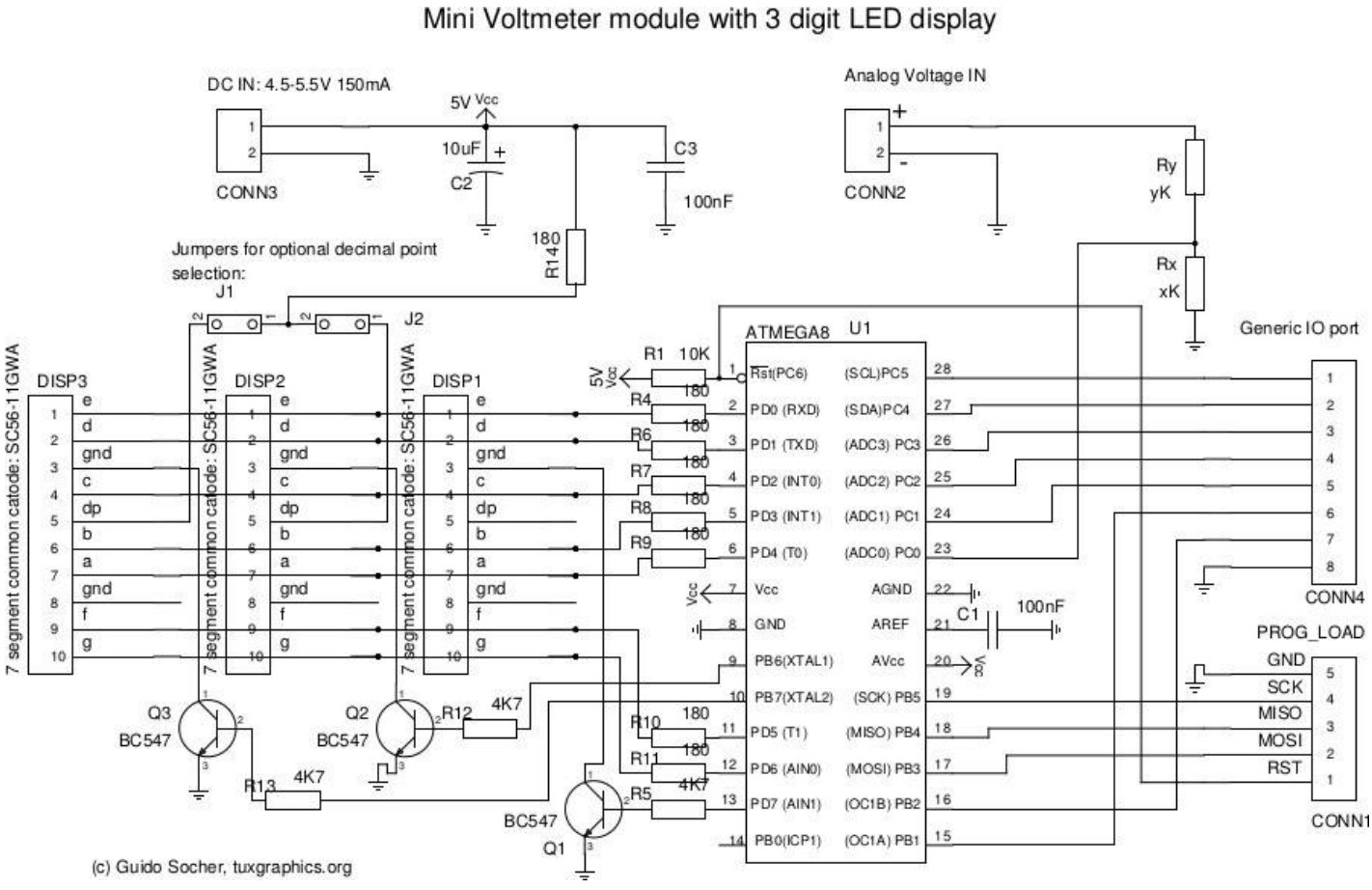


x=200µs y=0,5V fréquence lue= 2,36kHz

la mesure de la première partie de l'impulsion est faussée par la mise en parallèle du générateur de courant, il faut prendre la seconde oscillation

Schéma

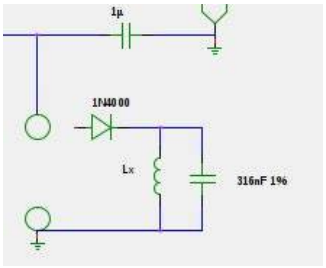




Avec l'aimable autorisation de G Socher

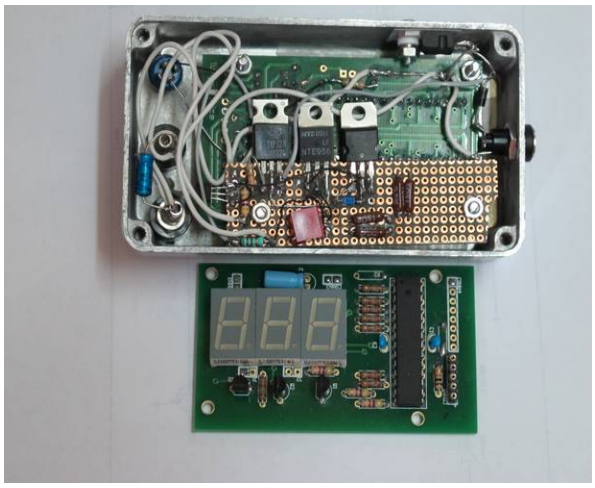
 zibuth27@gmx.fr www,hackerschicken.eu

La version finale utilise un montage Kelvin (4 fils) avec séparation des fils de génération de courant et des fils de mesure. Cela implique que les groupes de deux fils ne soient reliés ensemble qu'à l'endroit à mesurer, surtout pas de fils communs (ne rigolez pas, j'ai vu des directeurs techniques faire la gaffe, immédiatement notée dans le compte-rendu par le client, pour faire rigoler tout le monde).

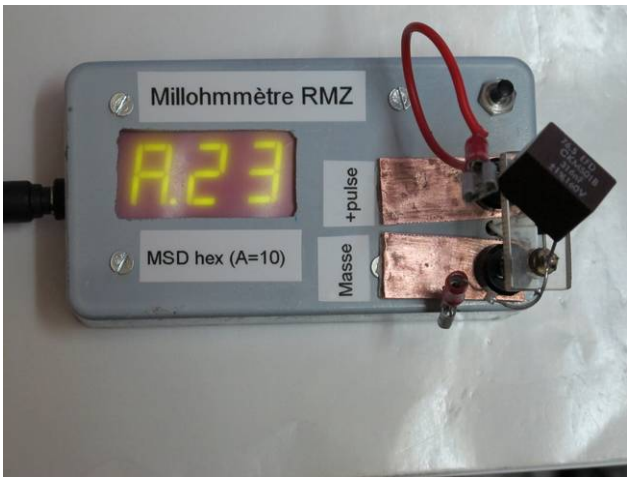


Adaptateur inductances monté sur les bornes de mesure

Réalisation



Le montage assemblé, en bas une carte Tuxgraphics neuve



montage assemblé, avec adaptateur inductances

Logiciel

Pas d'include particulier, les analog.c et analog.h de Tuxgraphics sont éliminés. Seules les bibliothèques standard AVR sont utilisées.

```
/*
* Milliohmmeter
* read tens of mohms
* Milliohmmeter by Zibuth27
* display tens of milliohm
* with parts from Guido Socher, Copyright: GPL 2
*
*
* Chip type           : Atmega8
* Clock frequency     : Internal clock 8 Mhz
*/
#include <avr/io.h>
#include <inttypes.h>
//#define F_CPU 8000000UL  // 8 MHz
#include <util/delay.h>


//----------------EDIT HERE-----------------------------------------
// where is the decimal point (needed in order to remove leading zeors):
// One digit behind decimal point:
//#define DP_DIG 1
// No decimal point:
#define DP_DIG 0
// Two digits after decimal point: (do not chop any zeros):
```

```
//#define DP_DIG 2
//-------------END EDIT HERE-----------------------------------------

//Connections of the seven segment elements:
//      10 9 7 6 4 2 1 = pin
//   0| -  f a b c d e | 0111111 | 3f
//   1| -  - - b c - - | 0001100 | 0c
//   2| g  - a b - d e | 1011011 | 5b
//   3| g  - a b c d - | 1011110 | 5e
//   4| g  f - b c - - | 1101100 | 6c
//   5| g  f a - c d - | 1110110 | 76
//   6| g  f a - c d e | 1110111 | 77
//   7| -  - a b c - - | 0011100 | 1c
//   8| g  f a b c d e | 1111111 | 7f
//   9| g  f a b c d - | 1111110 | 7e
//   A/ g  f a b c - e / 1111101 / 7d              // added hex figures for extending half a digit
//   b/ g  f - - c d e / 1100111 / 67
//   C/ -  f a - - d e / 0110011 / 33
//   d/ g  - - b c d e / 1001111 / 4f
//   E/ g  f a - - d e / 1110011 / 73
//   F/ g  f a - - - e / 1110001 / 71
// digit to seven segment LED mapping:
static                                                  unsigned                                                        char
d2led[]={0x3f,0x0c,0x5b,0x5e,0x6c,0x76,0x77,0x1c,0x7f,0x7e,0x7d,0x67,0x33,0x4f,0x73,0x71};


void init7segment(void)
{
        // least significant digit, cathode transistor:
        DDRD|= (1<<DDD7);
        // middle digit:
        DDRB|= (1<<DDB6);
        // left digit:
        DDRB|= (1<<DDB7);
        // digit pins:
        DDRD|= 0x7F;
        // off
        PORTD=0;
}

// Tuxgraphics code
// Update the sevent segement display.
// You must loop over this function all the time.
inline void upd7segment(unsigned int number)
{
        // LS digit:
        PORTD=d2led[(number%10)];
        //
        PORTD|=(1<<PD7);// digit on for a short time:
        _delay_ms(1);
        number/=10; // divide by 10 (shift off right most digit)
        PORTD&=~(1<<PD7);
#if (DP_DIG == 0)
        if (number>0){ // chop off the twp left most digits if it is zero
#endif
                // middle digit:
                PORTD=d2led[(number%10)];
                //
                PORTB|= (1<<PB6);// digit on for a short time:
                _delay_ms(1);
                number/=10; // divide by 10 (shift off right most digit)
                PORTB &= ~(1<<PB6);
#if (DP_DIG == 1 || DP_DIG == 0)
                if (number>0){ // chop off left most digit if it is zero
#endif
                        //
                        // left digit, we assume there that the number is
                        // smaller than 999, no modulo here:
                        PORTD=d2led[number];
                        //
                        PORTB|= (1<<PB7);// digit on for a short time:
                        _delay_ms(1);
                        PORTB &= ~(1<<PB7);
#if (DP_DIG == 1 || DP_DIG == 0)
                }
#endif
#if (DP_DIG == 0)
        }
```

```
#endif
}
// end of Tuxgraphics code

int main(void)                        // zibuth part : milliohmmeter
{
       uint16_t val;
        uint32_t  zero=0;
       uint8_t lc=0;          // loop counter

        init7segment();
       ADCSRA=(1<<ADEN)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);
       DDRC=(0<<PC5)|(1<<PC4)-(1<<PC3);
       PORTC=(1<<PC5);

        while (1) {
lc++;
if(lc==1){             // get into only part time
  ADMUX=0xc0;          // internal ref MUX0

    PORTC=(1<<PC3);    // current generator enable
                       // current pulse settling time
    if(bit_is_clear(PINC,PC5)){_delay_ms(50);} // long duration for inductive resistors
         else {_delay_ms(2);}

    ADCSRA|=(1<<ADSC);//start conversion
      while(bit_is_set(ADCSRA,ADSC)); // wait for result
       val=ADC;

    PORTC=(0<<PC3);    // curent generator inhibit


      val=1*val/1;
       if(val>1600)val=1599;
 //      val=555;


} // end if lc loop
lc=lc& 0x1f;

      upd7segment(val);              // loop almost always
}      //end while
        return(0);
}
```

le code compilé en format Intel :

```
:1000000012C024C023C022C021C020C01FC01EC0F7
:100010001DC01CC01BC01AC019C018C017C016C014
:1000200015C014C013C011241FBECFE5D4E0DEBF3D
:10003000CDBF10E0A0E6B0E0EAE4F1E002C0059038
:100040000D92A037B107D9F70AD07DC0D9CF8F9ACA
:10005000BE9ABF9A81B38F6781BB12BA0895F7DF4A
:1000600087E886B988E084BB80E285BB40E010EC7D
:1000700008E02AE030E04F5F4130D9F417B905BB02
:100080009D9905C083ED90E30197F1F704C083EFDC
:1000900091E00197F1F700C00000369A3699FECF43
:1000A000C4B1D5B115BAC13496E0D90710F0CFE389
:1000B000D6E04F71CE01B90132D0FC01E05AFF4FBA
:1000C000808182BB979A89EF90E00197F1F700C099
:1000D000000097986115710571F2CB01B9011FD02D
:1000E000FC01E05AFF4F808182BBC69A89EF90E005
:1000F0000197F1F700C00000C6986115710509F479
:10010000BACFFB01E05AFF4F808182BBC79A89EFCB
:1001100090E00197F1F700C00000C798ACCFAA1B90
:10012000BB1B51E107C0AA1FBB1FA617B70710F0E2
:10013000A61BB70B881F991F5A95A9F78095909514
:0A014000BC01CD010895F894FFCF33
:10014A003F0C5B5E6C76771C7F7E7D67334F7371E5
:00000001FF
```

Table des matières

 zibuth27@gmx.fr www,hackerschicken.eu