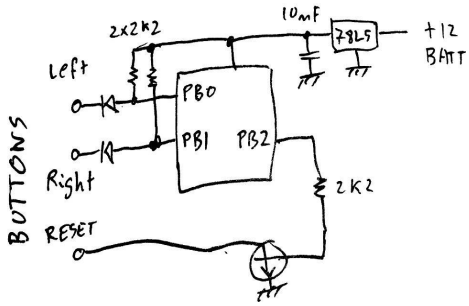
Modification des clignotants BMW, à la Harley

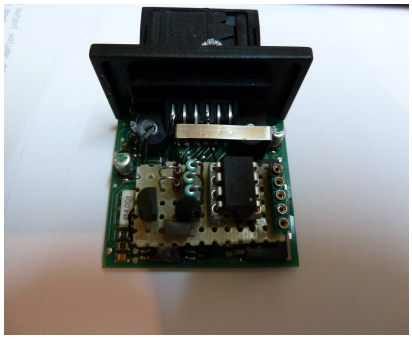
Clignotants à la Harley

J'ai eu la chance de pouvoir aller plusieurs fois à Las Vegas pour mon boulot. Quelques collègues ont cru que j'étais devenu essayeur de machines à sous, mauvaises langues! Plutôt que de louer une BMW et faire un tour dans le désert, j'ai préféré essayer une Harley-Davidson. Il ne faut pas mourir idiot et savoir pourquoi on préfère la BM.

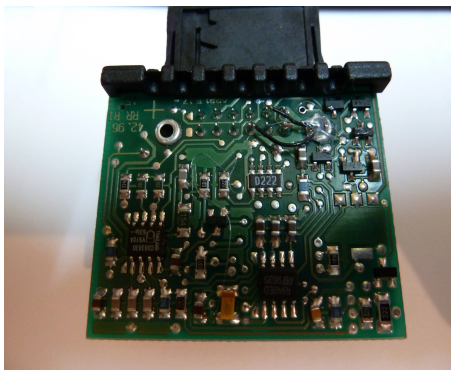
Je préfère toujours le confort R1100 RT, son freinage sa fourche antiplongée, son moteur qui n'est pas fait de 100kg de fonte, ça chauffe sacrément les cuisses dans le désert ! Et l'attente aux feux rouges est pénible. Mais la commande de clignotant est bien plus confortable : deux palettes comme sur la BM mais un coup ça marche, un coup ça s'arrête. Sur la BM ils on été inventer un bouton de reset qui s'encrasse vite, et il faut un mouvement totalement antinaturel, et parfois forcer, quel plaisir en hiver! J'ai donc implanté le mode Harley dans ma BM. Pour ne pas affoler un improbable autre utilisateur, j'ai laissé l'infâmeux bouton. Un petit microcontrôleur rajouté dans le boîtier clignotant fait le boulot. Qu'est-ce que c'est plus confortable maintenant ! Le montage doit être KISS (Keep It Simple Stupid)



Le schéma, très simple (KISS)



le microcontrôleur rajouté



les fils rajoutés

Le microcontrôleur AT Tiny 13 d'Atmel est un microcontrôleur RISC 8 bits travaillant ici en horloge interne 9,6MHz (pas d'horloge ni de composants externes) qui coute moins de 2€ par poignée de 10.

- développement avec des outils gratuits sous Linux/Ubuntu (gratuit lui aussi)
- on compile le c avec avr-gcc
- on programme la puce avec avrdude et un câble de programmation qui peut être réduit à 3 résistances sur port parallèle

Modification des clignotants BMW, à la Harley

```
/*
* BMW flasher modification, use like Harley Davidson: second push resets flasher
* At tiny13
*
*Zibuth27 31/07/2010
*/

#include <avr/io.h>
#include <util/delay.h>

#define F_CPU 96000000UL  // 9,6 MHz


/*************
PB0    p5     input left button
PB1    p6     input right button
PB2    p7     ouput reset button
***************/

int main()
{
  char left, right;

  DDRB = 0xFC;           // set direction to be output
  PORTB = 0x03;

while(1)
{
      if(bit_is_clear(PINB,PINB0))
      {
            switch (left)
            {
                  case 0:
                  {
                        while(bit_is_clear(PINB,PINB0))_delay_ms(10);
                        left=1;
                        break;
                  }
                  case 1:
                  {
                        PORTB|=_BV(PB2);
                        _delay_ms(40);    // reset pulse duration
                        PORTB&=~_BV(PB2);
                        left=0;right=0;
                        while(bit_is_clear(PINB,PINB0))_delay_ms(30);
                        break;
                  }

            }
      }
       if(bit_is_clear(PINB,PINB1))
      {
            switch (right)
            {
                  case 0:
                  {
                        while(bit_is_clear(PINB,PINB1))_delay_ms(10);
                        right=1;
                        break;
                  }
                  case 1:
                  {
                        PORTB|=_BV(PB2);
                        _delay_ms(40);    // reset pulse duration
                        PORTB&=~_BV(PB2);
                        right=0;left=0;
                        while(bit_is_clear(PINB,PINB1))_delay_ms(30);
```

```
                    break;
                }

            }
        }

 //  PB4 is a test flag, indicates any one flasher is active left or right

     if (left==1) PORTB|=_BV(PB4);
     else PORTB&=~_BV(PB4);
     if (right==1) PORTB|=_BV(PB4);
     else PORTB&=~_BV(PB4);

 // cleanup
     if(left>1)left=0;
     if(left<0)left=0;
     if(right>1)right=0;
     if(right<0)right=0;
}

}//end main
```

le code compilé, en mode Intel, prêt à rentrer dans la puce:

```
:1000000009C021C020C01FC01EC01DC01CC01BC015
:100010001AC019C011241FBECFE9CDBF10E0A0E661
:10002000B0E0EEE0F1E002C005900D92A036B1071D
:10003000D9F710E0A0E6B0E001C01D92A036B107EC
:10004000E1F702D062C0DCCF8CEF87BB83E088BBD6
:1000500060EC7DE540EF50E0B09921C0332331F0F2
:100060003130E9F407C0CB010197F1F7B09BFBCF2A
:1000700031E015C0C29A80E991E0FA013197F1F7B9
:100080000197D9F7C29807C08CE291E0FA01319745
:10009000F1F70197D9F7B09BF7CF30E020E0B199A5
:1000A00020C0222331F02130E1F406C0CB010197BA
:1000B000F1F7B19BFBCF14C0C29A80E991E0FA013D
:1000C0003197F1F70197D9F7C29807C08CE291E018
:1000D000FA013197F1F70197D9F7B19BF7CF05C036
:1000E00021E0313021F4C49A03C030E020E0C4980C
:1000F000213011F4C49A01C0C498323008F030E0C5
:0E010000223008F4A9CF20E0A7CFF894FFCF5B
:00000001FF
```