Electronique de pompe de relevage cave

Dans ma cave, les effluents des machines à laver, du bac à linge et du lavabo sont recueillis dans un trou de 30x30x30 cm dans lequel se trouve une pompe de relevage (400W) pour forcer l'eau à rejoindre le circuit d'évacuation.

Le trou est trop petit pour pouvoir utiliser l'interrupteur à flotteur livré avec la pompe. Celui-ci est bloqué en position haute, son étanchéité n'est donc pas modifiée.

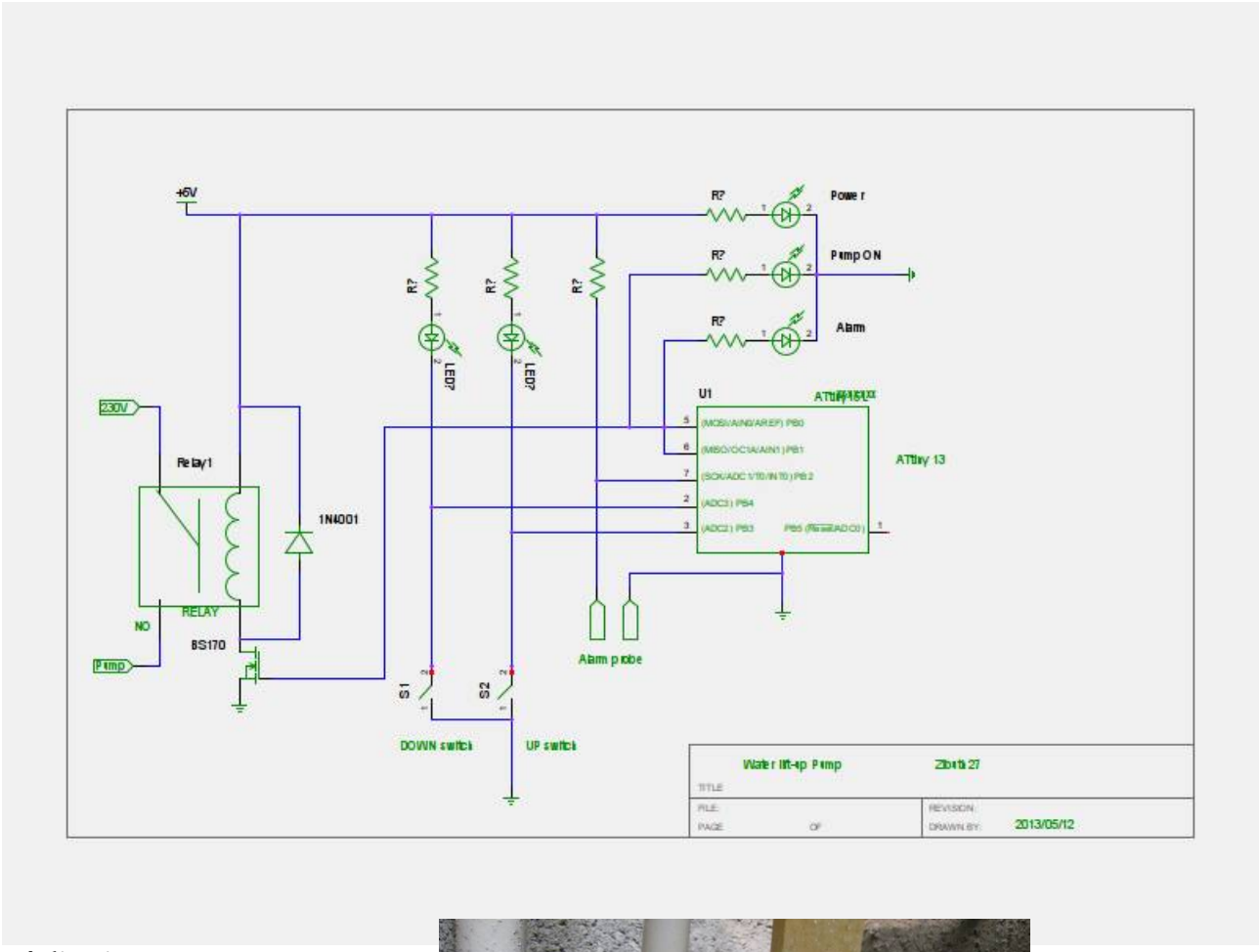
Capteurs
Les capteurs choisis sont des interrupteurs à lame souple (ILS) pour leur durée de vie (100 millions de coupures sur charge non inductives). Un courant de décrassage non inductif est assuré par l'allumage de LEDs jaunes à 20mA qui indiquent aussi le passage de l'aimant. L'ILS permet une certaine distance (jusqu'à 15mm entre l'aimant et l'ILS), et travaille sans contact physique.

Logique
Le circuit logique est assuré par un microcontroleur Attiny13 qui concentre en une puce la logique de détection/sécurité et les timers.

Fonctionnement
Un aimant est mis dans un tube ICO de 16mm, le tube est collé à un flotteur en tube PVC de 32mm. Deux crochets refermés assurent un guidage sur une structure en bois.
Lorsque l'aimant passe devant l'ILS du haut, la pompe est mise en route.
Lorsque l'aimant arrive à l'ILS du bas, une temporisation de 7s est mise en route pour prolonger le fonctionnement de la pompe car il faut vider plus, pour laisser ensuite se vider le tuyau vertical, qui retourne à la cuve. On évite ainsi un pompage permanent (l'autre solution possible est d'utiliser un clapet anti-retour mais au prix d'une perte de charge). Le temps de fonctionnement de la pompe est signalé par une LED orange.
Un capteur supplémentaire de sécurité est fait par deux lame d'inox de 4x300mm (provenant de balais d'essuie-glace) placées plus haut que le niveau de l'ILS du haut. Le contact est établi quand l'eau touche les deux électrodes. Le tirage au +5V est fait par une résistance de 330kohms, ce qui fait un courant d'électrolyse de 10μA, on détruira ainsi 26g de fer en 2 millions d'heures de contact avec l'eau. En fait, le circuit déclenche par capacité au contact de la première électrode avec l'eau, avant même que la seconde électrode ne touche.
Lorsque le circuit de sécurité est activé, la pompe est lancée pour 10 secondes, la LED rouge est activée.
L'indication de bon fonctionnement est assurée par le clignotement à 1Hz/100ms de la LED de sécurité.
Lorsque la sécurité est activée, le clignotement continue, en négatif car ce clignotement est fait par une fonction OU exclusif.
La commande de pompe est assurés par un MOSFET et un relais (on pourrait mettre un triac avec un opto)

Le circuit fonctionne normalement avec la détection par les ILS. En tout état de cause, l'alarme par détection ohmique intervient en cas de cuve pleine.
Comme il s'agit d'un circuit de sécurité (pour éviter l'inondation de la cave) le watchdog est utilisé. Il remet le programme à zéro si aucun reset de watchdog n'intervient avant 8 secondes (après le dernier).
Les machines à laver sont des générateurs de parasites importants qui perturbent parfois le microcontrôleur, tout comme la commande de pompe par relais.

Schéma



Réalisation

06/2013

Programme

```
/*****************************************
*  cellar lift pump
*
* Chip type           : AT tiny 13
* V2.0  2013/06/18 Zibuth27
* keywords : interrupt, probe water sense, watchdog
*
* this is a security programm (no willing to get cellar flooded), hence use of watchdog
*
* Clock frequency     : Internal clock 1.2 Mhz (normal delivery)
*
* status : complete
*****************************************/
#include <avr/io.h>
#include <util/delay.h>
#include <avr/sfr_defs.h>
#include <avr/wdt.h>
#include <avr/interrupt.h>

/*
t13 pinout & netlist
1      ISP pin5 RST
2      555    PB3    LO reed
3      555    PB4    HI reed
4      GND
5      PB0    pump out            MOSI
6      PB1    alarm out           MISO   Tx data from UART
7      PB2    emergency sense             SCK
8      Vcc
*/

volatile uint8_t cycles;

//initialize watchdog
void WDT_Init(void)
{
//disable interrupts
cli();
//reset watchdog
wdt_reset();
//set up WDT interrupt
WDTCR = (1<<WDCE)|(1<<WDE);
//Start watchdog timer with 8s prescaller
WDTCR = (1<<WDTIE)|(1<<WDE)|(1<<WDP3)|(1<<WDP0);
//Enable global interrupts
sei();
}


ISR (TIM0_OVF_vect)
{

      cycles++;
       if (cycles>20){                 // "alive" indicator
         PORTB^=(1<<PB1);
         _delay_ms(100);
         PORTB^=(1<<PB1);
         cycles=0;
       }
}


int main(void)
```

```
{
       DDRB=0x03;
       TCCR0A=0x00;
       TCCR0B=0x04;              // timer0 used, no output on any PORTB pin
       TIMSK0=(1<<TOIE0);

       sei();

          //initialize watchdog
       WDT_Init();

       while(1)
       {
  // alarm
       if(bit_is_clear(PINB,PB2))
       {
         PORTB|=(1<<PB0)|(1<<PB1);
         _delay_ms(10000);
       }
       else {PORTB&=(0<<PB0)|(0<<PB1);}


  // test positions
       if(bit_is_clear(PINB,PB4)){                  // HI switch
         PORTB|=(1<<PB0);
             loop_until_bit_is_clear(PINB,PB3);     // LO switch
              _delay_ms(7000);
              PORTB&=(0<<PB0);
       }
       else  PORTB&=(0<<PB0);
       //reset watchdog
       wdt_reset();
       }
}
```

code compilé format Intel

```
:1000000009C016C015C01DC013C012C011C010C059
:100010000FC00EC011241FBECFE9CDBF10E0A0E677
:10002000B0E001C01D92A136B107E1F734D05FC046
:10003000E7CFF894A89588E181BD89E681BD7894E1
:1000400008951F920F920FB60F9211248F939F93D2
:10005000EF93FF93809160008F5F809360008091A9
:1000600060008531 78F088B392E0892788BBEFE2A1
:10007000F5E73197F1F700C0000088B3892788BB06
:1000800010926000FF91EF919F918F910F900FBEA2
:100090000F901F90189583E087BB1FBC84E083BF3F
:1000A00082E089BF7894C5DFB2990DC088B38360C0
:1000B00088BB2FEF8EE994E2215080409040E1F719
:1000C00000C0000002C088B318BAB4990CC0C09A2E
:1000D000B399FECF2FE782EA99E12150804090400A
:1000E000E1F700C0000088B318BAA895DDCFF894F6
:0200F000FFCF40
:00000001FF
```