GEOVANNE ALMEIDA DE OLIVEIRA
RODRIGO MARTINS

MANUAL DE UTILIZAÇÃO
Dispositivo de controle de fluxo

UFSC - Blumenau

Instalação

O equipamento pode ser ligado ao encanamento diretamente ou receber água de fontes como galões e semelhantes. Para que o sistema de controle funcione, o equipamento deve estar ligado a rede elétrica, caso contrário, o dispositivo apenas permitirá a passagem de água ao acioná-lo manualmente. AVISO: Caso o sistema de controle seja ativado sem uma fonte de líquido disponível, a vida útil do dispositivo será reduzida, no pior caso, o pressurizador queimará.

Utilização

Configure no regulador quantos mililitros gostaria de encher, essa configuração permanecerá até que seja manualmente alterada. As cores do led RGB correspondem a:

Verde - 300ml;
Amarelo - 500ml;
Laranja - 700ml;
Vermelho - 1000ml;

Ao posicionar um copo no lugar, o dispositivo irá liberar a quantidade de líquido definida em sua configuração, parando automaticamente ao atingi-la. Caso o sensor ultrassônico identifique que o objeto ali posicionado não se trata de um copo, ele não permitirá a liberação do fluxo. Caso sejam detectados vazamentos, o sistema também automaticamente irá parar a operação, evitando desperdícios e risco ao usuário.

Vulnerabilidades e Melhorias

Futuramente os dispositivos poderão vir com filtros internos, compactando o sistema completo, além disso, criar uma versão portátil, com funcionamento ligado a uma bateria interna recarregável.

Código Fonte

```
#include <LiquidCrystal.h>
#include <avr/interrupt.h>

const byte pin_sensor = (1<<3);
const byte pin_trig = (1<<5);
```

```
const byte pin_echo = (1<<4);
int pin_red = 13;
int pin_green = 2;
int pin_blue = 6;
bool interrup = false;
LiquidCrystal lcd_1(12, 11, 10, 9, 8, 7);
void setup()
{
        DDRD |=(1<<5);
        DDRD |=(0<<4);
        lcd_1.begin(16, 2);
        Serial.begin(9600);
        cli();
        //CONFIG INTERRUPPCAO
        DDRD &= ~pin_sensor;
        PORTD |= pin_sensor;
        EIMSK |= (1<<INT1);
        EICRA |= (1<<ISC11) | (1<<ISC10);
        //CONFIG TIMER
        TCCR1A = 0;
        TCCR1B = 0;
        TCNT1  = 0;
        TCCR1B |= (1 << WGM12);
        OCR1A = 62499;
        TCCR1B |= (1 << CS12) | (1 << CS10);
        TIMSK1 |= (1 << OCIE1A);
        sei();
        //ADCONVERT
        ADCSRA= 0x93;
}

ISR(INT1_vect){
  lcd_1.clear();
  lcd_1.print("Cntrl de Fluxo");
  lcd_1.setCursor(0,1);
  lcd_1.print("ATIVADO");
  interrup = true;
  //Serial.println(calcAD());
}
```

```
ISR(TIMER1_COMPA_vect){
  lcd_1.clear();
  if(interrup)
    printDisplay(calcAD());
  interrup = false;
}

void printDisplay(int convAD){
        if(convAD<=255){
        if(getDistancia()<30){
                lcd_1.print("Recip. de 300ml");
                lcd_1.setCursor(0,1);
                lcd_1.print("Fluxo Minimo");
        }
        } else if(convAD<=511) {
        if(getDistancia()<30){
                lcd_1.print("Recip. de 500ml");
                lcd_1.setCursor(0,1);
                lcd_1.print("Fluxo em 50%");
        }
    } else if(convAD<=767){
        if(getDistancia()<30){
                lcd_1.print("Recip. de 700ml");
                lcd_1.setCursor(0,1);
                lcd_1.print("Fluxo em 75%");
        }
    } else {
        if(getDistancia()<30){
                lcd_1.print("Recip. de 1000ml");
                lcd_1.setCursor(0,1);
                lcd_1.print("Fluxo Maximo");
        }
    }
}

void loop()
{
  if(getDistancia()>200){
    lcd_1.clear();
        lcd_1.print("VAZAMENTO");
```

```
        lcd_1.setCursor(0,1);
        lcd_1.print("DETECTADO");
  }
  //if(getDistancia()>30&&getDistancia()<200){
  //      lcd_1.clear();
  //}
  if(calcAD()<=255){
        analogWrite(13,0);
        analogWrite(2,255);
        analogWrite(6,0);
        } else if(calcAD()<=511) {
        analogWrite(13,255);
        analogWrite(2,255);
        analogWrite(6,0);
    } else if(calcAD()<=767){
        analogWrite(13,255);
        analogWrite(2,140);
        analogWrite(6,0);
    } else {
        analogWrite(13,255);
        analogWrite(2,0);
        analogWrite(6,0);
    }
}

int calcAD(){
        ADMUX = 0x40;
        int analogH,analogL,analog;
        ADCSRA |= (1<<ADSC);
        while(!(ADCSRA&=~(1<<ADIF)));
        ADCSRA |=(1<<ADIF);
        analogL=ADCL;
        analogH=ADCH;
        analog=(analogH<<8)|analogL;
        return analog;
}

int getDistancia(){
        int tempo;
        float distancia;
```

```
        digitalWrite(pin_trig, LOW);
        delayMicroseconds(5);
        digitalWrite(pin_trig, HIGH);
        delayMicroseconds(10);
        digitalWrite(pin_trig, LOW);
        tempo = pulseIn(pin_echo, HIGH);
        distancia = tempo*170.00/10000;
        return distancia;
}
```