



Politechnika Wrocławska

Laboratorium konstrukcji urządzeń automatyki.

Semestr letni 2006/2007

Pulsometr.

Prowadzący:

mgr inż. Waldemar Sienkiewicz

Wykonali:

Karol Kozłowski 132652

Mateusz Zarzeczny 132945

DATA WYKONANIA:
15.06.2007

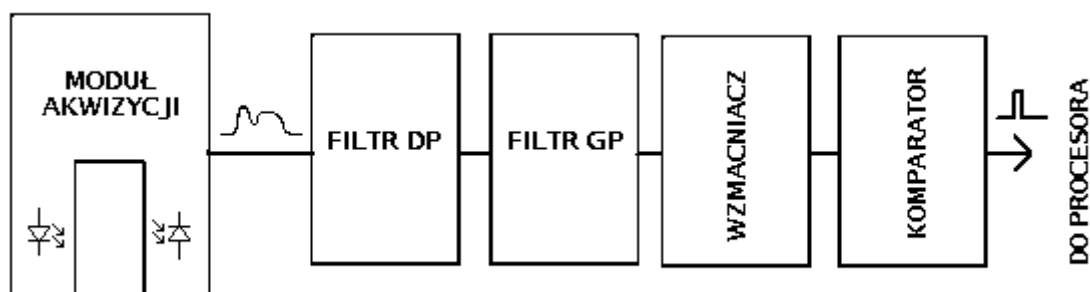
OCENA:

PODPIS:

1. Idea rozwiązania.

Rozwiązanie naszego układu chcieliśmy zrealizować maksymalnie prosto i maksymalnie skutecznie. Działanie opiera się na różnicy właściwości krwi natlenionej i nienatlenionej. Krew natleniona cechuje się lepszą przepuszczalnością promieni świetlnych, natomiast krew nienatleniona gorszą. Przepływ krwi oraz zmiany jej przepuszczalności odbywają się w rytm bicia serca. Jest to dobry modulator wiązki świetlnej, która po przejściu przez palec także zmienia swoje natężenie w rytm pulsu człowieka.

2. Zasada działania.



Światło wytworzyliśmy za pomocą kilku diod SMD wysokiej jasności – czerwonych. Odbiornikiem był układ optyczny z wbudowanym wzmacniaczem. Te dwa elementy: diody i detektor, powinny zostać umieszczone na przeciwległych stronach palca, w celu modulacji wiązki światła.

Drugim blokiem naszego układu jest filtr dolnoprzepustowy, którego zadaniem jest eliminować zakłócenia pochodzące z otoczenia, głównie od sieci 50Hz. Było to zadanie ułatwione, ponieważ częstotliwość główna pulsu ludzkiego dochodzi do maksymalnie 3Hz (w skrajnym przypadku). Choć ta różnica częstotliwości nie jest zbyt wielka, jednak w zupełności wystarczył do filtr RC pierwszego rzędu.

Kolejnym blokiem układu jest filtr górnoprzepustowy. Jego zadaniem jest odfiltrowywanie składowej stałej sygnału. Celem tego bloku jest eliminacja wpływu jasności świecenia diod oraz przepuszczalności świetlnej badanego. Jest także konieczna ze względu na niski stosunek amplitudy zmodulowanego pulsu do składowej stałej.

Blok wzmacniacza ma za zadanie „rozciągnąć” wcześniej przygotowany sygnał na szerszą skalę napięć. Celem takiego zabiegu jest przygotowanie sygnału do zamiany na sygnał cyfrowy, gdyż zwiększenie amplitudy zmian ułatwia dobranie odpowiedniego progu komparatora.

Ostatni blok – komparator ma za zadanie zamienić wzmacniony i odfiltrowany sygnał pulsu na postać cyfrową. Próg przełączania dobraliśmy tak, aby pojedynczy puls (składający się z dwóch „pików”) zamienić za pojedynczy impuls prostokątny.

Tak przetworzony sygnał jest gotowy do przekazania na część cyfrową układu.

3. Schemat ideowy. Schemat PCB.

Układ oparliśmy o układ LM324. Zawiera on 4 wzmacniacze operacyjne, z możliwością niesymetrycznego zasilania, które tutaj wykorzystaliśmy. Jako czujnik wykorzystaliśmy układ OPT101, który zawiera fotodiode, oraz wzmacniacz/separator sygnału.

Schematy ideowy i wzór płytki, dostępne są na końcu tego sprawozdania

4. Program procesora.

```
#define F_CPU 4000000 //czestotliwość pracy zegara
#define CYCLES_PER_US ((F_CPU+500000)/1000000) //tików na sekunde

#define LED PORTB //takie tam
#define LED_D DDRB
#define CTRL PORTC
#define CTRL_D DDRC
#define IN IN_I
#define IN_P PORTD
#define IN_I PIND
#define IN_D DDRD

#define TIM 65535-10000 // autoładowanie do timera
#include <avr/io.h> // obsługa we/wy
#include <util/delay.h> // obsługa opóźnień
#include <avr/interrupt.h> // obsługa przerwań

unsigned int t, meas, pulse, pulse1, pulse2, pulse3;

SIGNAL (SIG_INTERRUPT0) // przerwanie od INT0 (wejście na 4 pinie)
{
    unsigned int temp, off; // off – max dopuszczalna zmiana pomiaru
    off = 20;
    TCNT1 = TIM; // ładuj do timera wartość początkową
    CTRL |= _BV(4); // zapal diodę pomiarową
    temp = (70 * 80) / t; // przelicz na impulsy / minute
    if (meas == 0) { // ignoruj pierwszy pomiar
        meas = 1;
    }
    else{ // pomiary >1

        if ( (temp - pulse < off || pulse - temp < off) && pulse3 != 0 ){
            // jeżeli pomiar nieróżni się o więcej niż 20 od średniej
            // i pulse3 zawiera pomiar (zostały wykonane 3 początkowe pomiary)
```

```

        pulse3 = pulse2;           // przechowuje poprzednie pomiary
        pulse2 = pulse1;           // j.w.
        pulse1 = temp;              // zapisuje bieżący pomiar

        pulse = (pulse1 + pulse2 + pulse3) / 3; // średnia z ostatnich 3ch
    }
    else {
        if (pulse3 == 0) {          // jeżeli nie pobrał 3ch pierwszych pomiarów
            pulse3 = pulse2;
            pulse2 = pulse1;
            pulse1 = temp;
            pulse = (pulse1 + pulse2 + pulse3) / 3;
        }
    }
}
t = 0;                             // zeruj liczbę przepełnień timera
_delay_loop_2(2000);                // czekaj ileś tam ;)
}

SIGNAL(SIG_OVERFLOW1)              // przerwanie od przepełnienia
{
    TCNT1 = TIM;                    // wpisz do timera wart początkowa
    t++;                             // zwiększ t – liczba przepełnień timera
}
// funkcja dekodująca DEC -> LED
unsigned int decode(unsigned int num) {
    if (num == 0)
        return _BV(1) + _BV(2) + _BV(3) + _BV(4) + _BV(5) + _BV(6);
    // np – jeżeli jest 0, zapal bity 1,2,3,4,5,6

    if (num == 1)
        return _BV(3) + _BV(4);

    if (num == 2)
        return _BV(2) + _BV(3) + _BV(0) + _BV(6) + _BV(5);

    if (num == 3)
        return _BV(2) + _BV(3) + _BV(0) + _BV(4) + _BV(5);

    if (num == 4)
        return _BV(1) + _BV(0) + _BV(3) + _BV(4);

    if (num == 5)
        return _BV(2) + _BV(1) + _BV(0) + _BV(4) + _BV(5);

    if (num == 6)
        return _BV(2) + _BV(1) + _BV(6) + _BV(5) + _BV(4) + _BV(0);

    if (num == 7)
        return _BV(2) + _BV(3) + _BV(4);

    if (num == 8)
        return _BV(0) + _BV(1) + _BV(2) + _BV(3) + _BV(4) + _BV(5) + _BV(6);

    if (num == 9)
        return _BV(0) + _BV(1) + _BV(2) + _BV(3) + _BV(4) + _BV(5);
    return 0;
}

```

```

// wyświetl na LED
void disp( unsigned int num ) {
    unsigned int delay = 1500; // opóźnienie
    _delay_loop_2(delay);      // opóźnienie
    CTRL |= _BV(0);            // przełącz na LED – jednostki
    LED = decode(num % 10);    // oblicz 1 cyfrę, dekoduj i wyslij na port
    CTRL &= _BV(0);            // "opuść" LED – jednostki
    _delay_loop_2(delay);
    CTRL |= _BV(1);            // przełącz na LED – dziesiątki
    LED = decode(((num - (num % 10)) % 100) / 10); // oblicz 2 cyfrę, dekoduj i wyslij na port
    CTRL &= _BV(1);            // ...
    if (num / 100 != 0) {      // jeżeli są jakieś setki
        _delay_loop_2(delay); // delay
        CTRL |= _BV(2);      // przełącz na LED – setki
        LED = decode(num / 100); // oblicz 3 cyfrę, dekoduj i wyslij na port
        CTRL &= _BV(2);      // ...
    }
}

int main( void ) // program główny
{
    t=0; // zlicza ilość przepełnień timera
    pulse = 0; // przechowuje wartość pulsu

    CTRL_D = LED_D= 0xff; // usawia porty jako wyjście
    IN_D = 0; // j.w. tylko wejście

    IN_P = 0xff; // podnosi wejścia do 1
    LED = 0; // zeruje wyjście na LED

    TIMSK = _BV(TOIE1); // włącza przerwanie od INT0 (pin 4)
    TCNT1 = TIM; // ładuje wartość początkową do timera
    TCCR1A = 0x00; // czasomierz bez dodatków
    TCCR1B = _BV(CS00); // taktowany F_CPU

    GIMSK = _BV(INT0); // włącz obsługę przerw
    MCUCR = _BV(ISC01); // INT0 reaguje na zbocze opadające

    sei(); // włącza obsługę przerw

    while(1) // pętla nieskończona (programu)
    {
        if (t > 150) {
            // jeżeli nie ma pomiaru przez dłuższy czas zeruj zmienne
            meas = pulse = pulse1 = pulse2 = pulse3 = 0;
        }
        if (pulse < 60 && pulse != 0) {
            // jeżeli puls poniżej 60 – bradykardia – zapal diodę
            CTRL |= _BV(3);
        }
        if (pulse > 100) {
            // jeżeli puls powyżej 100 – tachykardia – zapal inną diodę
            CTRL |= _BV(5);
        }
        disp(pulse); // wyświetl tętno
    }
}
//KONIEC

```

Powyższy program po uruchomieniu rozpoczyna odczytywanie sygnałów (przychodzący sygnał wywołuje zewnętrzne przerwanie) wysyłanych przez moduł analogowy. Pierwszy pomiar zostaje zignorowany. Następne 3 ładowane są do zmiennych i na ich podstawie obliczana jest średnia wartość tętna. Kolejne odczytywane wartości porównywane są z wartością średnią i jeżeli bieżący pomiar różni się o 20 od średniej jest uznawany jako błędny i ignorowany.

Wyznaczona wartość pulsu zostaje podzielona na setki dziesiątki i jedności przy użyciu funkcji `display()`. Następnie funkcja `decode()` zamienia liczbę dziesiętną na format wyświetlacza siedmio segmentowego i wysyła tę wartość na port LED. 3 cyfrowy wyświetlacz jest skonstruowany w z wykorzystaniem multipleksacji – każda cyfra wyświetlana jest po kolei – anody wyświetlaczy są połączone a katody załączane są poprzez tranzystory sterowane wyjściami mikrokontrolera.