UNIWERSYTET ZIELONOGÓRSKI

Wydział Elektrotechniki, Informatyki i Telekomunikacji

CYFROWE PRZETWARZANIE SYGNAŁÓW Z ZASTOSOWANIEM PROCESORÓW SYGNAŁOWYCH - PROJEKT

REGULATOR PID Z POMIAREM WILGOTNOŚCI ORAZ TEMPERATURY

Korneliusz Mietlicki

1. Opis urządzenia i zasada działania

Urządzenie zbudowane w ramach projektu z przedmiotu Cyfrowe przetwarzanie sygnałów z zastosowaniem procesorów sygnałowych, powstało w celu sterowania grzałkami teleskopów astronomicznych przy wykonywaniu astrofotografii oraz obserwacji, w celu uniknięcia pokrywania się rosą optyki, w warunkach dużej wilgotności względnej powietrza. W celu optymalnej pracy temperatura optyki musi być utrzymywana na poziomie nieznacznie przekraczającym temperaturę punktu rosy, zależną od wilgotności względnej powietrza oraz temperatury, co zapobiegnie roszeniu optyki, zminimalizuje pobór energii z akumulatorów, a także zredukuje negatywne efekty falowania powietrza z powodu dużej różnicy temperatur na granicy optyka-powietrze. „Sercem" układu jest 8 bitowy mikrokontroler Microchip PIC16F1827, pracujący z zegarem o częstotliwości 16MHz, który został wybrany ze względu na wystarczającą pojemność pamięci (7KB); wbudowany stabilny generator zegarowy z pętlą PLL; sprzętowy interfejs I^2C; cztery sprzętowe 10-bitowe kanały PWM, możliwość generacji przerwania zewnętrznego z dowolnego wyjścia PORTU B. Pomiary temperatury otoczenia oraz wilgotności względnej wykonywane są przez cyfrowy czujnik Sensirion SHT21 z którym łączność odbywa się za pomocą interfejsu I^2C, natomiast temperatura obiektu mierzona jest za pomocą cyfrowego czujnika temperatury Maxim Dallas DS1624, połączonego za pomocą tej samej magistrali I^2C co czujnik wilgotności, umożliwiło to zredukowanie do minimum ilość przewodów pomiędzy sterownikiem a czujnikami. Parametry odczytane z czujników, oraz ustawiona przez użytkownika zadana temperatura obiektu stanowią parametry wejściowe regulatora PID, natomiast parametrem wyjściowym tego regulatora jest sygnał PWM o rozdzielczości 10 bitów, sterujący tranzystory MOSFET IRF540. Za grzanie obiektów odpowiadają grzałki składające się z rezystorów o łącznej mocy ok. 30W każda. Prezentacja aktualnie odczytanych, obliczonych oraz nastawionych parametrów realizowana jest na podświetlanym wyświetlaczu 2x16 znaków, opartym na popularnym sterowniku HD44780. Nad działaniem procesora czuwa układ WatchDog, który w przypadku jego zawieszenia lub zapętlenia się generuje sygnał resetu, co zapobiega np. nadmiernemu grzaniu obiektu, co mogłoby doprowadzić np. do szybkiego rozładowania się akumulatorów.

2. Pomiary wilgotności oraz temperatur

Wilgotność względna powietrza mierzona jest przez czujnik Sensirion SHT21, o rozdzielczości pomiaru 12 bitów i dokładności ±2%. Transmisja danych odbywa się za pomocą dwukierunkowej, dwuprzewodowej magistrali I²C z prędkością 400kb/s. Ze względu na dużą wartość błędu pomiaru wilgotności, jedynie jej część całkowita jest wykorzystywana do dalszych obliczeń. Czas pomiaru wilgotności wynosi średnio 22ms.

Temperatura otoczenia jest mierzona również przez czujnik SHT21. Pomiar temperatury odbywa się z rozdzielczością 14bitów i dokładności ±0,3ºC, podobnie jak w przypadku wilgotności do obliczeń wykorzystywana jest wyłącznie część całkowita zmierzonej temperatury. Czas pomiaru temperatury wynosi średnio 66ms.

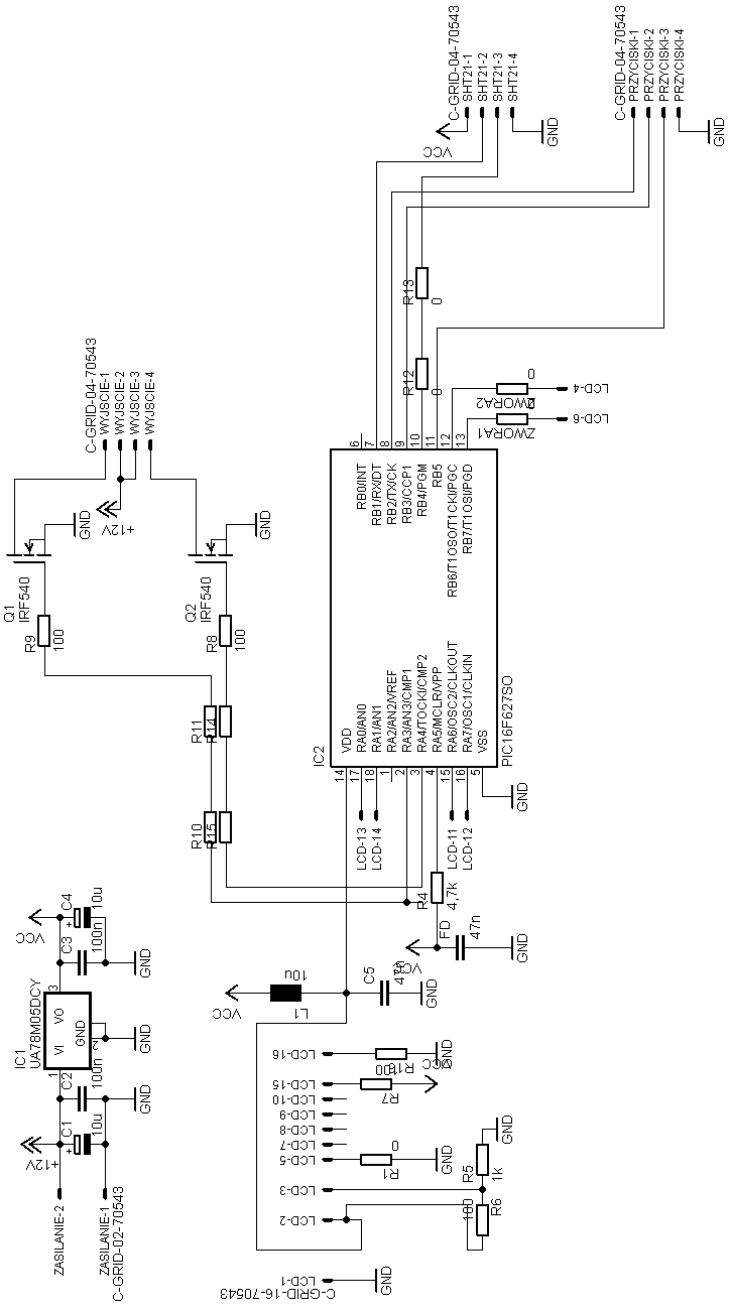
Czujnikiem temperatury obiektu jest Maxim Dallas DS1624 o rozdzielczości 13 bitów i dokładności pomiaru ±0,5ºC. W celu uzyskania dużej stabilności temperatury obiektu, jako parametr wejściowy regulatora PID wykorzystywana jest pełna, 13 bitowa rozdzielczość czujnika. Czas pomiaru przez czujnik wynosi około 400ms.

3. Budowa oraz schematy układu

Układ został zaprojektowany w całości za pomocą oprogramowania Eagle w wersji 5.10. Jako źródło zasilania części cyfrowej użyty został stabilizator liniowy LM78M05, z kondensatorami zarówno na jego wyjściu jak i wejściu które zapobiegają wzbudzaniu się stabilizatora. Sam procesor oraz wyświetlacz są dodatkowo zabezpieczone dławikiem na zasilaniu, co pozwoliło zmniejszyć prawdopodobieństwo przypadkowego zawieszenia się procesora przy sterowania grzałkami o dużych mocach, które mogłyby generować zakłócenia. Elementami sterującymi są tranzystory IRF540, o typowej rezystancji dren-źródło na poziomie 0,055Ω, co przy prądach grzałek na poziomie do 3A umożliwiło zrezygnowanie z radiatorów. Bramki tranzystorów sterowane są przez rezystory o wartości 100Ω bezpośrednio z wyjściami kanałów sprzętowych generatorów sygnału PWM. Do zmiany parametrów urządzenia użyte zostały przyciski zwierne, łączące piny

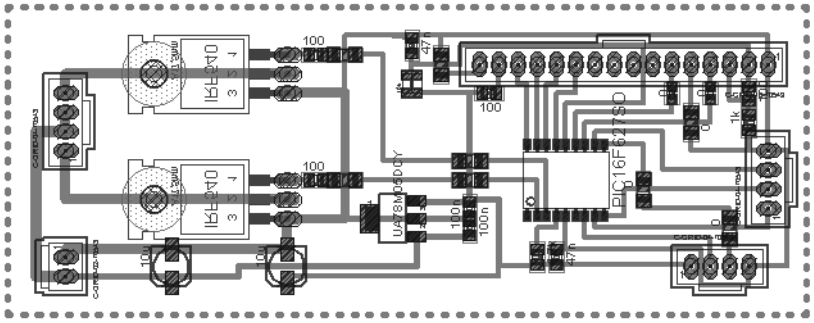
mikrokontrolera z masą. Piny wejściowe podciągane są za pomocą wewnętrznych rezystorów „pull-up” do napięcia zasilającego. W celu uzyskania dużej niezawodności oraz małej wrażliwości na zakłócenia zasilania zrezygnowano z wewnętrznego podciągana pinu ‘RESET’ na rzecz zewnętrznego układu RC na tym pinie.

3.1. Schemat ideowy



Rysunek 1. Schemat ideowy

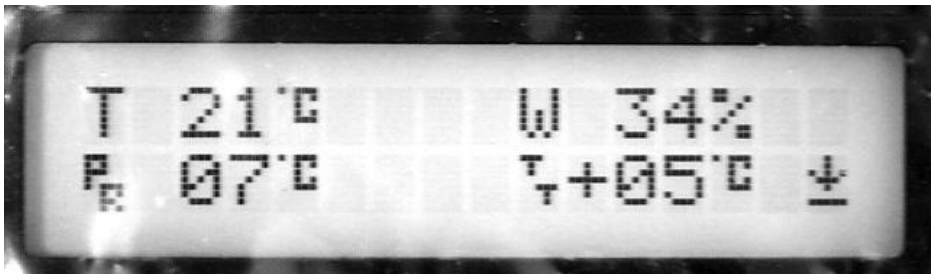
3.2. Schemat montażowy



Rysunek 2. Schemat montażowy płytki

3.3. Wyświetlacz LCD

Dane na wyświetlaczu prezentowane są w następujący sposób:



Rysunek 3. Widok wyświetlacza sterownika

- T – temperatura otoczenia w ºC
- PR – temperatura punktu rosy w ºC
- W – wilgotność względna powietrzna w %
- TT – zadana temperatura teleskopu w ºC powyżej temperatury punktu rosy
- Symbol grzania/chłodzenia lub właściwej temperatury teleskopu

4. Oprogramowanie

4.1. Środowisko programistyczne

Mikrokontroler został zaprogramowany w języku C, w środowisku MikroC PRO for PIC, które zostało wybrane ze względu na dużą ilość bibliotek wspomagających programowanie ułatwiających obsługę wyświetlacza LCD, interfejsu I^2C, generatora sygnału PWM, czy też układu WatchDog, a także na szeroką gamę obsługiwanych procesorów z rodziny PIC oraz dsPIC. Skompilowany program zajął 98% pamięci programu oraz 36% pamięci danych procesora.

4.2. Kod programu

```
int x;
short int funkcja;
int time;

char *text = " 00.0";
char *temp1 = " 00";
char *wilgotnosc = " 00%";
char *pkt_rosy = " 00";
unsigned temp;
const char character1[] = {23,5,4,5,7,0,0,0}; //Deklaracja symbolu stopni Celsiusza
const char tt[] = {28,8,8,0,7,2,2,0}; //Deklaracja symbolu temperatury teleskopu
const char pr[] = {28,20,28,16,23,5,6,5}; //Deklaracja symbolu temperatury rosy
const char ok[] = {0,10,10,0,17,14,0,0}; //Deklaracja symbolu właściwej temperatury
const char hi[] = {4,14,21,4,4,0,31,0}; //Deklaracja symbolu grzania teleskopu
const char lo[] = {4,4,21,14,4,0,31,0}; //Deklaracja symbolu ochładzania teleskopu

//Funkcja wyświetlająca symbol ochładzania teleskopu w wyznaczonym miejscu na LCD
void t_lo(char pos_row, char pos_char) {
  char i;
    Lcd_Cmd(120);
    for (i = 0; i<=7; i++) Lcd_Chr_CP(lo[i]);
    Lcd_Cmd(_LCD_RETURN_HOME);
    Lcd_Chr(pos_row, pos_char, 7);
}

//Funkcja wyświetlająca symbol grzania teleskopu w wyznaczonym miejscu na LCD
void t_hi(char pos_row, char pos_char) {
  char i;
    Lcd_Cmd(112);
    for (i = 0; i<=7; i++) Lcd_Chr_CP(hi[i]);
    Lcd_Cmd(_LCD_RETURN_HOME);
    Lcd_Chr(pos_row, pos_char, 6);
}

//Funkcja wyświetlająca symbol właściwej temperatury teleskopu na LCD
void buzka_ok(char pos_row, char pos_char) {
  char i;
    Lcd_Cmd(96);
    for (i = 0; i<=7; i++) Lcd_Chr_CP(ok[i]);
    Lcd_Cmd(_LCD_RETURN_HOME);
    Lcd_Chr(pos_row, pos_char, 4);
}

//Funkcja wyświetlająca symbol temperatury punktu rosy na LCD
void punkt_rosy(char pos_row, char pos_char) {
  char i;
    Lcd_Cmd(80);
    for (i = 0; i<=7; i++) Lcd_Chr_CP(pr[i]);
    Lcd_Cmd(_LCD_RETURN_HOME);
    Lcd_Chr(pos_row, pos_char, 2);
}

//Funkcja wyświetlająca symbol temperatury teleskopu na LCD
```

```
void t_teleskopu(char pos_row, char pos_char) {
  char i;
    Lcd_Cmd(88);
    for (i = 0; i<=7; i++) Lcd_Chr_CP(tt[i]);
    Lcd_Cmd(_LCD_RETURN_HOME);
    Lcd_Chr(pos_row, pos_char, 3);
}

//Funkcja wyświetlająca symbol stopni Celsiusza na LCD
void znak_stopni(char pos_row, char pos_char) {
  char i;
    LCD_Cmd(64);
    for (i = 0; i<=7; i++) LCD_Chr_Cp(character1[i]);
    LCD_Cmd(_LCD_RETURN_HOME);
    LCD_Chr(pos_row, pos_char, 0);
}


// Struktura zawierająca parametry regulatora PID //

struct {
signed int Tp; //temperatura obiektu
signed int zadana; //temperatura zadana obiektu 1
signed long int uchyb; // uchyb
signed long int uchyb_pop; //poprzednia temperatura obiektu
short unsigned int Kp; //wzmocnienie członu proporcjonalnego
short unsigned int Ki; //wzmocnienie członu całkującego
short unsigned int Kd; //wzmocnienie członu różniczkującego
long signed int out; //wyjście
signed long int P,I,D; //obliczone wartości współczynników PID
signed int I_pop; //poprzednia wartość współczninnika członu całkującego
int out_min; //minimalna wartość wyjścia
int out_max; //maksymalna wartość wyjścia

int DDelay;
int IDelay;
int wzrost;
} PID1;


// Koniec struktury zawierającej parametry regulatora PID //

//Deklaracja portów wyjściowych służących do sterowania wyświetlacza LCD
sbit LCD_RS at RB6_bit;
sbit LCD_EN at RB7_bit;
sbit LCD_D4 at RA6_bit;
sbit LCD_D5 at RA7_bit;
sbit LCD_D6 at RA0_bit;
sbit LCD_D7 at RA1_bit;

sbit LCD_RS_Direction at TRISB6_bit;
sbit LCD_EN_Direction at TRISB7_bit;
sbit LCD_D4_Direction at TRISA6_bit;
sbit LCD_D5_Direction at TRISA7_bit;
sbit LCD_D6_Direction at TRISA0_bit;
```

```
sbit LCD_D7_Direction at TRISA1_bit;
//Koniec deklaracji portów wyświetlacza LCD



//Funkcja przeliczająca temperaturę odczytaną z czujnika DS1624 na stopnie Celsiusza
long int Display_Temperature(unsigned int temp2zapis,unsigned x, unsigned y) {
  char temp_whole;
  unsigned int temp_fraction;

 //sprawdzenie czy temperatura nie jest ujemna, dodanie znaku
if (temp2zapis & 0x8000) {
    text[0] = '-';
    temp2zapis = ~temp2zapis + 1;
    }
    else
    {
    text[0] = ' ';
    }

  // Wyciągnięcie części całkowitej temperatury
  temp_whole = temp2zapis >> 8 ;
  // Zamienia część całkowitą temperatury na pojedyncze cyfry
  text[1] = (temp_whole/10)%10 + 48;           // dziesiątki
  text[2] =  temp_whole%10     + 48;           // jedności stopni

  // Wyciągnięcie i zamiana na INT'a części ułamkowej
  temp_fraction  = temp2zapis>>3; //usuwa puste bity
  temp_fraction &= 0x001F;   //usuwa liczby całkowite z temeratury
 temp_fraction *= 3.125;    //krok temperatury

  // Zamienia część  część ułamkową na pojedyncze cyfry
  text[4] =  temp_fraction/10   + 48;        // tysiące

  //Lcd_Out(x, y, text);
// znak_stopni(x,y+5);

return ((temp_whole*100)+(temp_fraction));
}

//Funkcja odczytu temperatury z czujnika DS1625 poprzez interfejs I2C
int czyt_temp(short int adres)
{
int s_bajt,m_bajt;
   I2C1_Start();
  I2C1_Wr(adres);        //Wybór czujnika z którego odczytywane są dane
  I2C1_Wr(0xAC);     //Komenda zapisu rejestru konfiguracyjnego
  I2C1_Wr(0x00);     //Wybór automatycznego pomiaru temeratury

  Delay_ms(10);            //Opóźnienie niezbędne do prawidłowej komunikacji z czujnikiem –
                           //komendy wysyłane do czujnika zapamiętywane są w jego pamięci
                           //EEPROM do której dostęp wymaga opóźnień

  I2C1_Repeated_Start();
  I2C1_Wr(adres);
```

```
 I2C1_Wr(0xEE);     //uruchomienie pomiaru
 I2C1_Stop();
 Delay_ms(10);

 I2C1_Start();
 I2C1_Wr(adres);
 I2C1_Wr(0xAA);     //polecenie odczytu temeratury
  Delay_ms(10);

 I2C1_Repeated_Start();
 I2C1_Wr(adres+1);
 s_bajt = I2C1_Rd(1);       //odczyt starszego bajtu temperatury
 m_bajt = I2C1_Rd(0);       //odczyt młodszego bajtu temperatury

 I2C1_Stop();

return (s_bajt<<8)+m_bajt; //zespolenie obu bajtów zawierających temperaturę obiektu
}

//Konfiguracja czujnika wilgotności i temperatury SHT21
void konfiguruj_czujnik()
    {
      I2C1_Start();
      I2C1_Wr(0x80);    //adres zapisu do czujnika SHT21
      I2C1_Wr(0xE7);    //komenda zapisu rejestru konfiguracyjnego
      I2C1_Wr(0x01);    //tryb RH 12bit/T 14 bit, grzałka wyłączona
      I2C1_Stop();
    }

//wyświetla względną wilgotność powietrza
void wyswietl_wilgotnosc(signed int wil, unsigned int x, unsigned int y)
{
 wilgotnosc[1]=((wil/10)%10)+48;
 wilgotnosc[2]=((wil)%10)+48;
 Lcd_Out(x,y,wilgotnosc);
}

  //wyświetla temperaturę, część całkowitą temperatury na wyświetlaczu w określonym przez
//parametry wejściowe położeniu.
void wyswietl_temperature(signed int temp_1, unsigned int x, unsigned int y{
 if(temp_1<0)
          {
          temp1[0]='-';
          }
 else
          {
          temp1[0]=' ';
          }

 temp_1=abs(temp_1);
 temp1[1]=((temp_1/10)%10)+48;
 temp1[2]=(temp_1%10)+48;

 Lcd_Out(x,y,temp1);
 znak_stopni(x,y+3);
```

```
}

//odczyt temperatury z czujnika SHT21
signed int odczyt_temperatury()
{
int s_bajt,m_bajt;
   I2C1_Start();
   I2C1_Wr(0x80);           //adres zapisu do czujnika SHT21
   I2C1_Wr(0xE3);           //komenda odczytu temperatury w trybie HOLD
   I2C1_Repeated_Start();
   I2C1_Wr(0x81);           //adres odczytu z czujnika SHT21
   s_bajt = I2C1_Rd(1);      //odczyt starszego bajtu temperatury
   m_bajt = I2C1_Rd(1);      //odczyt młodszego bajtu temperatury
   I2C1_Rd(0);              // suma kontrolna CRC - pominięto
   I2C1_Stop();

   m_bajt&=0x02;
   s_bajt=s_bajt<<8;

return (-46.85+((175.72*(s_bajt+m_bajt))/65536));

}

//odczyt wilgotności z czujnika SHT21
unsigned int odczyt_wilgotnosci()
{
int s_bajt,m_bajt;

   I2C1_Start();
   I2C1_Wr(0x80);           //adres zapisu do czujnika SHT21
   I2C1_Wr(0xE5);           //komenda odczytu wilgotności w trybie HOLD
   I2C1_Repeated_Start();
   I2C1_Wr(0x81);           //adres odczytu z czujnika SHT21
   s_bajt = I2C1_Rd(1);      //odczyt starszego bajta wilgotności
   m_bajt = I2C1_Rd(1);      //odczyt młodszego bajta wilgotności
   I2C1_Rd(0);              // suma kontrolna CRC - pominięto
   I2C1_Stop();

   m_bajt&=0x02;
   s_bajt=s_bajt<<8;

 return (-6+(125*(s_bajt+m_bajt))/65536);;
 }

//obliczenie punktu rosy na podstawie temperatury i wilgotności odczytanych z czujnika SHT21
signed int oblicz_punkt_rosy()
{
return
((0.0016*(odczyt_wilgotnosci())+0.8413)*(112+(0.9*odczyt_temperatury()))+(0.1*odczyt_temperatury())-112);  // obliczanie punktu rosy
}
```

```
//funkcja dzieląca wartość wejściową na starsze i młodsze bity w celu zapisania do rejestrów
//generatora sygnału PWM
void PWM_10bit(int z)
 {
 CCPR3L=z>>2;
 CCP3CON |= ((z&0b00000011)<<4);

  CCPR4L=z>>2;
 CCP4CON |= ((z&0b00000011)<<4);
 }


//Funkcja obsługi przerwania zewnętrznego pochodzącego z przycisków
 void interrupt() {
asm {CLRWDT}; //kasowanie licznika układu WatchDog

if( Button(&PORTB, 2, 50, 1)) //jesli zostanie naciśnięty przycisk '+'
{
   if(PID1.wzrost<30) //sprawdź czy nie przekroczono zakresu
       {
       PID1.wzrost++; //inkrementuj wartość

       }
   else
       {
       PID1.wzrost=30; //jeśli przekroczono zakres przypisz maksymalną wartość
       }

}

if( Button(&PORTB, 3, 50, 1)) //jeśli zostanie naciśnięty przycisk '-'
{
    if(PID1.wzrost>1)
       {
              PID1.wzrost--;
       }
    else
       {
       PID1.wzrost=1;
       }

}
EEPROM_Write(0x01, PID1.wzrost); //po ustawieniu wartości przez użytkowinika zapisz ją
//w pamięci EEPROM mikrokontrolera

asm {CLRWDT};
INTCON.IOCIF=0;    //kasowanie globalnej flagi przerwań
IOCBF.IOCBF2=0;    //kasowanie flagi przerwań portu B.2
IOCBF.IOCBF3=0;    //kasowanie flagi przerwań portu B.3
}



void main() {
 signed long int ds1624_1;
```

```
 int idelay,ddelay;


//Ustawienie zegara procesora na 16MHz z pętlą PLL //
   OSCCON.IRCF3=1;
   OSCCON.IRCF2=1;
   OSCCON.IRCF1=1;
   OSCCON.IRCF0=0;
   OSCCON.SPLLEN=0;

//Ustawienie portów A i B jako porty cyfrowe //
   ANSELA=0x00;
   ANSELB=0x00;

  TRISA=0x00;
  TRISB=0x00;

//Globalne zezwolenie na podciąganie portów wejściowych //
  OPTION_REG &=(0<<7);

//Ustawienie pinów B2, B3 i B5 jako wejściowe //
  TRISB.B2=1;
  TRISB.B3=1;
  TRISB.B5=1;

//Włączenie wewnętrznego podciągania do pinów B2, B3 i B5 //
  WPUB.WPUB2=1;
  WPUB.WPUB3=1;
  WPUB.WPUB5=1;

//Konfiguracja przerwań do pinów wejściowych B2, B3, B5 //
            INTCON.IOCIE=1;
            INTCON.GIE=1;

            IOCBN.IOCBN2=1;
            IOCBN.IOCBN3=1;

 asm {CLRWDT};
I2C1_Init(50000);
konfiguruj_czujnik();

    Lcd_Init();                         // Initialize LCD

  Lcd_Cmd(_LCD_CLEAR);                  // Clear LCD
  Lcd_Cmd(_LCD_CURSOR_OFF);
  Lcd_Out(1,1,"Blad czujnikow!");
  asm {CLRWDT};
  Delay_ms(250);
  asm {CLRWDT};
  Lcd_Cmd(_LCD_CLEAR);


  PWM3_Init(1000);
  PWM3_Start();
```

```
 PWM4_Init(1000);
 PWM4_Start();


//Parametry regulatora PID1 //

  PID1.wzrost=EEPROM_Read(0x01); //odczytaj z pamięci EEPROM poprzednio
//ustawiona temperature zadana

  PID1.Kp = 35; //wzmocnienie członu proporcjonalnego
  PID1.Ki = 1; //wzmocnienie członu całkującego
  PID1.Kd = 40; //wzmocnienie członu różniczkującego

  PID1.I=0; //przypisanie początkowej wartości całki
  PID1.out_min=0; //minimalna wartość wyjściowa sygnału PWM
  PID1.out_max=1023; //maksymalna wartość wyjściowa sygnału PWM
  PID1.IDelay=20; //opóźnienie członu całkującego
  PID1.DDelay=20; //opóźnienie członu różniczkującego

  ddelay=PID1.DDelay;
  idelay=PID1.IDelay;

    while(1)
{
 asm {CLRWDT};



Lcd_Out(1,1,"T");
wyswietl_temperature(odczyt_temperatury(),1,2); //wyświetlenie temperatury otoczenia
asm {CLRWDT};

Lcd_Out(1,10,"W");
wyswietl_wilgotnosc(odczyt_wilgotnosci(),1,11); //wyświetlenie wilgotności otoczenia
asm {CLRWDT};

punkt_rosy(2,1);
wyswietl_temperature(oblicz_punkt_rosy(),2,2); //wyświetlenie temperatury punktu rosy
asm {CLRWDT};
t_teleskopu(2,10);

  wyswietl_temperature(PID1.wzrost,2,11); //wyświetlenie zadanej temperatury
 Lcd_Out(2,11,"+");

//obliczenie temperatury wejściowej zadanej dla regulatora PID
 PID1.zadana=((oblicz_punkt_rosy()*100)+PID1.wzrost*100)/3.124;
 if(PID1.P<16 && PID1.P > -16)
 {
 buzka_ok(2,16); //wyświetlenie symbolu właściwej temperatury, gdy różnica między
//temperaturą obiektu, a temperaturą zadaną jest mniejsze od 0,5°C
 }
 else
 {
  if(PID1.P>16)
          {
```

```
        t_hi(2,16); //jeśli temperature teleskopu jest wyższa o conajmniej 0,5ºC wyświetl
                //symbol chłodzenia teleskopu
            }
  if(PID1.P<-16 )
             {
            t_lo(2,16); //wyświetlenie symbolu grzania teleskopu
            }
 }

asm {CLRWDT};



 ds1624_1=czyt_temp(0x92); //odczyt temperatury z czujnika DS1624
  ds1624_1 = Display_temperature(ds1624_1,2,11); //przeliczenie temperatury

  if(text[0]=='-') //sprawdzenie czy temperature jest ujemna, jesli tak wyświetlenie symbolu '-
'
            {
            ds1624_1*=-1;
            }

PID1.uchyb_pop=PID1.uchyb;  //zapamiętanie poprzedniej wartości temperatury na cele
całkowania
PID1.I_pop=PID1.I; //zapamiętanie poprzedniej wartości całki

 PID1.Tp=ds1624_1/3.124;

PID1.uchyb=PID1.zadana-PID1.Tp; //obliczenie uchybu temperatury

PID1.P=PID1.uchyb; //Przypisanie uchybu do zmiennej członu proporcjonalnego

PID1.I=((PID1.uchyb+PID1.uchyb_pop)/2)+PID1.I_pop;  //obliczenie  całki  metodą
//trapezów i przypisanie wyniku do zmiennej członu całkującego

//instrukcje ograniczające wartość zmiennej członu całkującego do 10 bitów.
if(PID1.I>256)
            {
            PID1.I=1023;
            }
if(PID1.I<1)
            {
            PID1.I=0;
            }

PID1.D=(PID1.uchyb-PID1.uchyb_pop); //obliczenie wartości członu różniczkującego

PID1.out=PID1.P*PID1.Kp+PID1.I*PID1.Ki+PID1.D*PID1.Kd;  //sumowanie  wartości
//członów regulatora z uwzględnieniem wzmocnień tych członów

//ograniczenie  wartości  wyjściowej  regulatora,  tak  aby  mieściła  się  ona  w  zakresie
//generatora sygnału PWM
if(PID1.out>=PID1.out_max)
            {
            PID1.out=PID1.out_max;
```

```
            }
if(PID1.out<=PID1.out_min)
            {
            PID1.out=PID1.out_min;
            }

 return PID1.out; //zwrócenie obliczonej wartości
}
.out=0;
            }
  PWM_10bit(PID1.out);

  asm {CLRWDT};
 }
}
```

5. Dobór parametrów regulatora PID

Dobór elementów regulatora przeprowadzono korzystając z metody Zieglera-nicholsa. Do wykonywania pomiarów użyto czujnik DS18B20 o rozdzielczości 12 bitów, sprzężony z oprogramowaniem na PC generującym w czasie rzeczywistym wykres zmiany temperatury. Pierwszym etapem było wyłączenie członów całkującego i różniczkującego regulatora, a następnie stopniowe zwiększanie wzmocnienia członu proporcjonalnego aż do wystąpienia oscylacji układu. Na podstawie wykresu temperatury podczas tętnień wyznaczono ich okres, co zostało następnie użyte do wyznaczenia wzmocnienia pozostałych członów na podstawie tabeli:

Tabela 1. Wyznaczanie parametrów regulatora PID

Kp	Ki	Kd
0,6 Ku	2*Kp / Tu	Kp*Tu / 8

Gdzie:

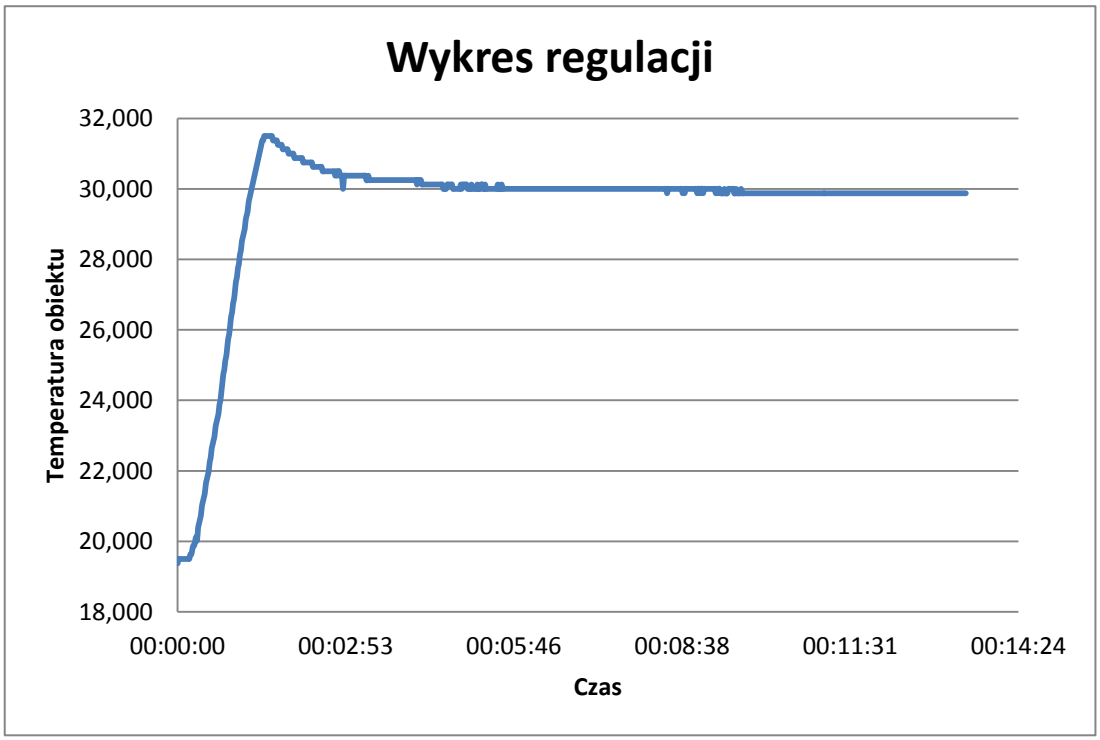
- Kp – wzmocnienie członu proporcjonalnego
- Ki – wzmocnienie członu całkującego
- Kd – wzmocnienie członu różniczkującego
- Ku – wzmocnienie członu proporcjonalnego przy którym występują oscylacje
- Tu – okres oscylacji przy wzmocnieniu Ku

Na tej podstawie wyznaczono następujące wzmocnienia poszczególnych członów:

- Człon proporcjonalny: Kp = 35;
- Człon całkujący: Ki = 1;
- Człon różniczkujący: Kd = 40;

6. Badania sterownika

Badanie sterownika zostało wykonane za pomocą czujnika DS18B20, tego samego który wykorzystywany był podczas doboru parametrów regulatora. Czujnik został skonfigurowany do pracy z maksymalną rozdzielczością 12 bitów, w celu rejestracji nawet niewielkich zmian temperatury obiektu. Pomiary temperatury wykonywane były na docelowym obiekcie z którym pracował będzie sterownik. Temperaturą zadaną było 30ºC. Różnica pomiędzy temperaturą zadaną, a temperaturą obiektu widoczną na wykresie, wynika z błędu czujników zarówno sterownika, jak i czujnika DS18B20 który wykorzystywany był do tworzenia wykresu. Przed włączeniem regulatora temperatura otoczenia oraz obiektu wynosiła 19,5ºC, w czasie T+10 sekund sterownik został włączony. W czasie T+3 minut błąd temperatury pomiędzy obiektem a temperaturą zadaną spadł poniżej wartości 0,5ºC, natomiast po czasie T+9 minut nastąpiła całkowita stabilizacja temperatury obiektu na poziomie 29,875ºC.



Wykres 1. Wykres regulacji temperatury sterownika