**Sprawozdanie**

*Systemy wbudowane*



**Ćwiczenie 8:**  
Czujniki – pomiar temperatury.

Wykonanie:

**Busłowski Tomasz**

**Suchwałko Tomasz**

**Skrouba Kamil**

**Zawadzka Magdalena  
(Grupa PS3)**

Prowadzący zajęcia: **dr inż. Adam Klimowicz**

Zakres Materiału

1. Wewnętrzny czujnik temperatury mikrokontrolera.
2. Czujnik temperatury STLM20.
3. Czujnik temperatury TC77.
4. Interfejs SPI.

Zadania do wykonania

1. Korzystając z przykładu 1 napisz program wyświetlający temperaturę procesora na dowolnym wyświetlaczu.
2. Korzystając z przykładu 2 napisz program wyświetlający temperaturę zmierzoną czujnikiem TC77 na dowolnym wyświetlaczu.
3. Napisz program wyświetlający temperaturę na wyświetlaczu LCD korzystając z czujnika wbudowanego w mikrokontroler i czujnika STLM20. Wyprowadzenie czujnika znajdziesz na złączu Con7 (Temp). Do przetwarzania A/C wykorzystaj tryb wielokanałowy przetwornika. Zdefiniuj symbol stopnia „°” na wyświetlaczu LCD.
4. Napisz program porównujący wskazania czujnika STLM20 dla 2 metod obliczania wartości temperatury (liniowej i nieliniowej).

Zadanie 1

**Treść:**

Korzystając z przykładu 1 napisz program wyświetlający temperaturę procesora na dowolnym wyświetlaczu.

**Realizacja:**

* Najpierw utowrzyliśmy projekt na podstawie przykładu 1.
* W tym zadaniu (i w nstępnym) wyświetlaliśmy temperaturę na wyświetlaczu alfanumerycznym LCD, więc podłączyliśmy wyświetlacz tak jak w Ćwiczeniu 5.
* Następnie kod programu z przykładu zmodyfikowaliśmy wykorzystaując kod   
  z Ćwiczenia 5. tak, aby wyświetlał odczytaną temperaturę na wyświetlaczu LCD.
* W celu konwersji wartości z **double** na **tablicę znaków** wykorzystaliśmy funkcję   
  **sprintf** z dodatkowo zaimportowanej biblioteki "cstdio".

**Kod programu – plik main.c:**

#include "stm32f10x.h"

#include "stm32f10x\_adc.h"

#include "lcd.h"

#include "cstdio"

void GPIO\_Config(void);

void RCC\_Config(void);

void NVIC\_Config(void);

void ADC\_Config(void);

int main(void)

{

volatile unsigned long int i;

unsigned long int wartoscADC1 = 0;

double result = 25.0;

unsigned char temperatura[5];

RCC\_Config();

GPIO\_Config();

ADC\_Config();

LCD\_Initialize(); // Inicjalizuj LCD

LCD\_WriteCommand(LCD\_DISPLAY\_ONOFF | LCD\_DISPLAY\_ON);

LCD\_WriteCommand(0x01); // Wyczysc LCD

while (1) {

ADC\_SoftwareStartConvCmd(ADC1, ENABLE); //wyzwolenie pojedynczego pomiaru

while (!ADC\_GetFlagStatus(ADC1,ADC\_FLAG\_EOC)); //odczekaj na zakonczenie konwersji

wartoscADC1 = ADC\_GetConversionValue(ADC1); //pobiez zmierzona wartosc

/\*Tu wstaw kod konwertuj¹cy wynik wg. Wzoru i prezentuj¹cy go na wyœwietlaczu\*/

result = (1430.0-wartoscADC1\*0.8067)/4.3 + 25.0;

sprintf(temperatura, "%f", result);

for (i=0; i< 5; i++) // Wypisz tekst LCD\_WriteData(temperatura[i]);

for (i=0;i<1000000ul;i++); //oczekanie

LCD\_WriteCommand(0x01);

};

while(1);

return 0;

}

void RCC\_Config(void)

//konfigurowanie sygnalow taktujacych

{

ErrorStatus HSEStartUpStatus; //zmienna opisujaca rezultat uruchomienia HSE

RCC\_DeInit(); //Reset ustawien RCC

RCC\_HSEConfig(RCC\_HSE\_ON); //Wlaczenie HSE

HSEStartUpStatus = RCC\_WaitForHSEStartUp(); //Odczekaj az HSE bedzie gotowy

if(HSEStartUpStatus == SUCCESS)

{

FLASH\_PrefetchBufferCmd(FLASH\_PrefetchBuffer\_Enable);//

FLASH\_SetLatency(FLASH\_Latency\_2); //ustaw zwloke dla pamieci Flash; zaleznie od taktowania rdzenia

//0:<24MHz; 1:24~48MHz; 2:>48MHz

RCC\_HCLKConfig(RCC\_SYSCLK\_Div1); //ustaw HCLK=SYSCLK

RCC\_PCLK2Config(RCC\_HCLK\_Div1); //ustaw PCLK2=HCLK

RCC\_PCLK1Config(RCC\_HCLK\_Div2); //ustaw PCLK1=HCLK/2

RCC\_PLLConfig(RCC\_PLLSource\_HSE\_Div1, RCC\_PLLMul\_9); //ustaw PLLCLK = HSE\*9 czyli 8MHz \* 9 = 72 MHz

//RCC\_PLLConfig(RCC\_PLLSource\_HSE\_Div1, RCC\_PLLMul\_7); //ustaw PLLCLK = HSE\*7 czyli 8MHz \* 7 = 56 MHz - konieczne dla ADC

RCC\_PLLCmd(ENABLE); //wlacz PLL

while(RCC\_GetFlagStatus(RCC\_FLAG\_PLLRDY) == RESET); //odczekaj na poprawne uruchomienie PLL

RCC\_SYSCLKConfig(RCC\_SYSCLKSource\_PLLCLK); //ustaw PLL jako zrodlo sygnalu zegarowego

while(RCC\_GetSYSCLKSource() != 0x08); //odczekaj az PLL bedzie sygnalem zegarowym systemu

/\*Tu nalezy umiescic kod zwiazny z konfiguracja sygnalow zegarowych potrzebnych w programie peryferiow\*/

RCC\_ADCCLKConfig(RCC\_PCLK2\_Div6); // ADCCLK = PCLK2/6 = 72 MHz /6 = 12 MHz

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA, ENABLE);//wlacz taktowanie portu GPIO A

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB, ENABLE);//wlacz taktowanie portu GPIO B

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_AFIO, ENABLE);//wlacz taktowanie portu AFIO

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_ADC1, ENABLE); //wlacz taktowanie ADC1

} else {

}

}

void GPIO\_Config(void)

{

//konfigurowanie portow GPIO

GPIO\_InitTypeDef GPIO\_InitStructure;

// disable JTAG

GPIO\_PinRemapConfig(GPIO\_Remap\_SWJ\_JTAGDisable, ENABLE);

/\*Tu nalezy umiescic kod zwiazany z konfiguracja poszczegolnych portow GPIO potrzebnych w programie\*/

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_0 | GPIO\_Pin\_1 ;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IN\_FLOATING;

GPIO\_Init(GPIOA, &GPIO\_InitStructure);

}

void ADC\_Config(void)

{

//konfigurowanie przetwornika AC

ADC\_InitTypeDef ADC\_InitStructure;

ADC\_InitStructure.ADC\_Mode = ADC\_Mode\_Independent; //Jeden przetwornik, praca niezalezna

ADC\_InitStructure.ADC\_ScanConvMode = DISABLE; //Pomiar jednego kanalu, skanowanie kanalow nie potrzebne

ADC\_InitStructure.ADC\_ContinuousConvMode = DISABLE; //Pomiar w trybie jednokrotnym

ADC\_InitStructure.ADC\_ExternalTrigConv = ADC\_ExternalTrigConv\_None; //Brak wyzwalania zewnetrznego

ADC\_InitStructure.ADC\_DataAlign = ADC\_DataAlign\_Right; //Wyrownanie danych do prawej - 12 mlodszych bitow znaczacych

ADC\_InitStructure.ADC\_NbrOfChannel = 1; //Liczba uzywanych kanalow =1

ADC\_Init(ADC1, &ADC\_InitStructure); //Incjalizacja przetwornika

ADC\_RegularChannelConfig(ADC1, ADC\_Channel\_16, 1, ADC\_SampleTime\_239Cycles5);

//Kanal 16 - wewnetrzny czujnik temp. procesora

//Grupa podstawowa, czas probkowania 239,5+12,5=252 cykli = 18us (calkowity czas przetwarzania)

//239,5 => 17.1us = czas zalecany w dokumentacji MUC dla miernika temperatury

ADC\_TempSensorVrefintCmd(ENABLE); //Wlaczenie czujnika temperatury w procesorze

ADC\_Cmd(ADC1, ENABLE); //Wlacz ADC1

ADC\_ResetCalibration(ADC1); //Reset rejestrow kalibracyjnych ADC1

while(ADC\_GetResetCalibrationStatus(ADC1)); //Odczekanie na wykonanie resetu

ADC\_StartCalibration(ADC1); //Kalibracja ADC1

while(ADC\_GetCalibrationStatus(ADC1)); //Odczekanie na zakonczenie kalibracji ADC1

}

Zadanie 2

**Treść:**

Korzystając z przykładu 2 napisz program wyświetlający temperaturę zmierzoną czujnikiem TC77 na dowolnym wyświetlaczu.

**Realizacja:**

* Realizaja zadania 2 polegała na modyfikacji zadania 1 – również wyświetlaliśmy temperaturę na wyświetlaczu alfanuerycznym LCD.
* Zgodnie z treścią przykładu 2 podłączymyśmy czujnik temperatury TC77: Con15 łączymy   
  z pinami złącza Con8 (PA4 → CS, PA5 → SCK, PA7 → SIO).
* Dodaliśmy do projektu moduł SPI z biblioteki StdPeriph i zaimportowaliśmy w pliku main.c.

**Kod programu – plik main.c (zmienione ciało funkcji main.c):**

#include "stm32f10x.h"

#include "stm32f10x\_adc.h"

#include "stm32f10x\_spi.h"

#include "lcd.h"

#include "cstdio"

void GPIO\_Config(void);

void RCC\_Config(void);

void NVIC\_Config(void);

void ADC\_Config(void);

void SPI\_Config(void);

int main(void)

{

volatile unsigned long int i;

unsigned long int wartoscADC1 = 0;

double result = 25.0;

unsigned char temperatura[5];

int temperatura2=12345;

RCC\_Config();

GPIO\_Config();

ADC\_Config();

RCC\_Config();

NVIC\_Config();

SPI\_Config();

LCD\_Initialize(); // Inicjalizuj LCD

LCD\_WriteCommand(LCD\_DISPLAY\_ONOFF | LCD\_DISPLAY\_ON);

LCD\_WriteCommand(0x01); // Wyczysc LCD

#define SPI\_Mode\_Slave\_Mask ((unsigned short int)0xFEFB) //Maska pozwalajaca wyzerowac bity trybu pracy wprost w rejestrzez SPIx->CR1

while (1) {

SPI1->CR1 |= SPI\_Mode\_Master; //Ustaw tryb master - wymusi to zmiane stanu NSS na niski

while (SPI\_I2S\_GetFlagStatus(SPI1, SPI\_I2S\_FLAG\_RXNE) == RESET); //Czekaj na dane

temperatura2 = SPI\_I2S\_ReceiveData(SPI1); //Odczytaj dane

if ((temperatura2&0x04)==0){ //Sprawdz, czy zakonczono juz pierwszy pomiar po wlaczeniu ukladu TC77

temperatura2=0; //Jesli nie, ustaw temp=0, wprzeciwnym razie wynik bedzie bledny (>500stC)

}

temperatura2 = temperatura2 >> 3; //Usun 3 LSB

SPI1->CR1 &= SPI\_Mode\_Slave\_Mask; //Ustaw tryb slave - wymusi to zmiane stanu NSS na wysoki

/\*Tu wstaw kod konwertuj¹cy wynik i prezentuj¹cy go np. na wyœwietlaczu\*/

sprintf(temperatura, "%d", temperatura2);

temperatura[3] = temperatura[2];

temperatura[2] = '.';

temperatura[4] = '0';

for (i=0;i<100000ul;i++);

for (i=0; i< 5; i++) // Wypisz tekst

LCD\_WriteData(temperatura[i]);

//for (i=0;i<1000000ul;i++);

for (i=0;i<4500000ul;i++); // opóŸnienie

LCD\_WriteCommand(0x01);

};

while (1);

return 0;

}

Zadanie 3

**Treść:**

Napisz program wyświetlający temperaturę na wyświetlaczu LCD korzystając z czujnika wbudowanego w mikrokontroler i czujnika STLM20. Wyprowadzenie czujnika znajdziesz na złączu Con7 (Temp). Do przetwarzania A/C wykorzystaj tryb wielokanałowy przetwornika. Zdefiniuj symbol stopnia „°” na wyświetlaczu LCD.

**Kod programu:**

void GPIO\_Config(void);

void RCC\_Config(void);

void NVIC\_Config(void);

void ADC\_Config(void);

void write\_temp(int temp);

int adc\_read(int channel);

int main(void) {

volatile unsigned long int i;

volatile char tekst1[10] = {"pomiar z: "};

volatile char tekst2[8] = {"stlm20: "};

volatile int temp = 0;

unsigned long int wartoscADC1 = 0;

//konfiguracja systemu

RCC\_Config();

GPIO\_Config();

ADC\_Config();

LCD\_Initialize(); // Inicjalizuj

LCD LCD\_WriteCommand(LCD\_CLEAR); // Wyczysc LCD

while (1) {

//odczyt wewnętrznego czujnika

wartoscADC1 = adc\_read(ADC\_Channel\_16);

temp = ((1430-(wartoscADC1\*0.8067))/4.3 + 25)\*10;

LCD\_WriteCommand( LCD\_DDRAM\_SET | 0x00 );

for (i=0; i<10; i++)

LCD\_WriteData(tekst1[i]);

LCD\_WriteCommand( LCD\_DDRAM\_SET | 0x08 );

write\_temp(temp);

//odczyt kanalu 0 gpio 0

wartoscADC1 = adc\_read(ADC\_Channel\_0);

LCD\_WriteCommand( LCD\_DDRAM\_SET | 0x28 );

for (i=0; i<8; i++)

LCD\_WriteData(tekst2[i]);

temp = 85.543\*(1.8663 - (wartoscADC1\*0.8067/1000))\*100;

write\_temp(temp);

for (i=0;i ;i<3200000ul; i++);

};

while (1);

return 0;

}

void write\_temp(int temp) {

LCD\_WriteData(48 + ((temp/1000)%10));

LCD\_WriteData(48 + (temp/100) % 10);

LCD\_WriteData(',');

LCD\_WriteData(48 + (temp/10)%10);

LCD\_WriteData(48 + temp%10);

LCD\_WriteData(0xdf); LCD\_WriteData('C');

}

void RCC\_Config(void)

//konfigurowanie sygnalow taktujacych

{

ErrorStatus HSEStartUpStatus; //zmienna opisujaca rezultat uruchomienia

HSE RCC\_DeInit(); //Reset ustawien RCC

RCC\_HSEConfig(RCC\_HSE\_ON); //Wlaczenie HSE

HSEStartUpStatus = RCC\_WaitForHSEStartUp(); //Odczekaj az HSE bedzie gotowy

if(HSEStartUpStatus == SUCCESS) {

FLASH\_PrefetchBufferCmd(FLASH\_PrefetchBuffer\_Enable); FLASH\_SetLatency(FLASH\_Latency\_2); //ustaw zwloke dla pamieci Flash; zaleznie od taktowania rdzenia //0:48MHz

RCC\_HCLKConfig(RCC\_SYSCLK\_Div1); //ustaw HCLK=SYSCLK RCC\_PCLK2Config(RCC\_HCLK\_Div1); //ustaw PCLK2=HCLK RCC\_PCLK1Config(RCC\_HCLK\_Div2); //ustaw PCLK1=HCLK/2

RCC\_PLLConfig(RCC\_PLLSource\_HSE\_Div1, RCC\_PLLMul\_9); //ustaw PLLCLK = HSE\*9 czyli 8MHz \* 9 = 72 MHz

RCC\_PLLCmd(ENABLE); //wlacz PLL

while(RCC\_GetFlagStatus(RCC\_FLAG\_PLLRDY) == RESET); //odczekaj na poprawne uruchomienie PLL

RCC\_SYSCLKConfig(RCC\_SYSCLKSource\_PLLCLK); //ustaw PLL jako zrodlo sygnalu zegarowego

while(RCC\_GetSYSCLKSource() != 0x08); //odczekaj az PLL bedzie sygnalem zegarowym systemu

RCC\_ADCCLKConfig(RCC\_PCLK2\_Div6); // ADCCLK = PCLK2/6 = 72 MHz /6 = 12 MHz

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA, ENABLE);//wlacz taktowanie portu GPIO A

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB, ENABLE);//wlacz taktowanie portu GPIO B

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_AFIO, ENABLE));//wlacz taktowanie portu AFIO

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_ADC1, ENABLE); //wlacz taktowanie ADC1

} else { }

}

void GPIO\_Config(void) {

//konfigurowanie portow GPIO

GPIO\_InitTypeDef GPIO\_InitStructure;

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_0|GPIO\_Pin\_1;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AIN;

GPIO\_Init(GPIOA, &GPIO\_InitStructure); // disable JTAG

GPIO\_PinRemapConfig(GPIO\_Remap\_SWJ\_JTAGDisable, ENABLE);

}

int adc\_read(int channel)

{

ADC\_RegularChannelConfig(ADC1, channel, 1, ADC\_SampleTime\_239Cycles5); ADC\_SoftwareStartConvCmd(ADC1, ENABLE);

while (!ADC\_GetFlagStatus(ADC1, ADC\_FLAG\_EOC));

return ADC\_GetConversionValue(ADC1);

}

void ADC\_Config(void) {

//konfigurowanie przetwornika AC

ADC\_InitTypeDef ADC\_InitStructure;

ADC\_InitStructure.ADC\_Mode = ADC\_Mode\_Independent; //Jeden przetwornik, praca niezalezna

ADC\_InitStructure.ADC\_ScanConvMode = DISABLE; //Pomiar jednego kanalu, skanowanie kanalow nie potrzebne

ADC\_InitStructure.ADC\_ContinuousConvMode = DISABLE; //Pomiar w trybie jednokrotnym

ADC\_InitStructure.ADC\_ExternalTrigConv = ADC\_ExternalTrigConv\_None; //Brak wyzwalania zewnetrznego

ADC\_InitStructure.ADC\_DataAlign = ADC\_DataAlign\_Right; //Wyrownanie danych do prawej - 12 mlodszych bitow znaczacych

ADC\_InitStructure.ADC\_NbrOfChannel = 1; //Liczba uzywanych kanalow =1

ADC\_Init(ADC1, &ADC\_InitStructure); //Incjalizacja przetwornika

ADC\_RegularChannelConfig(ADC1, ADC\_Channel\_16, 1, ADC\_SampleTime\_239Cycles5); //Kanal 16 - czujnik temperatury procesor

ADC\_Cmd(ADC1, ENABLE); //Wlacz ADC1

ADC\_ResetCalibration(ADC1); //Reset rejestrow kalibracyjnych ADC1 while(ADC\_GetResetCalibrationStatus(ADC1)); //Odczekanie na wykonanie resetu ADC\_StartCalibration(ADC1); //Kalibracja ADC1

while(ADC\_GetCalibrationStatus(ADC1)); //Odczekanie na zakonczenie kalibracji ADC1

}

Zadanie 4

**Treść:**

Napisz program porównujący wskazania czujnika STLM20 dla 2 metod obliczania wartości temperatury (liniowej i nieliniowej).

**Kod programu:**

void GPIO\_Config(void);

void RCC\_Config(void);

void NVIC\_Config(void);

void ADC\_Config(void);

void write\_temp(int temp);

int adc\_read(int channel);

int main(void) {

volatile unsigned long int i;

volatile char tekst1[8] = {"linia: "};

volatile char tekst2[10] = {"nielinia: "};

volatile int temp = 0;

unsigned long int wartoscADC1 = 0; //konfiguracja systemu

RCC\_Config();

GPIO\_Config();

ADC\_Config();

LCD\_Initialize(); // Inicjalizuj LCD

LCD\_WriteCommand(LCD\_CLEAR); // Wyczysc LCD

while (1) {

//odcyzt kanalu 0, gpio 0

wartoscADC1 = adc\_read(ADC\_Channel\_0);

temp = 85.543\*(1.8663 - (wartoscADC1\*0.8067/1000))\*100;

LCD\_WriteCommand( LCD\_DDRAM\_SET | 0x00 );

for (i=0; i<8 ;i++)

LCD\_WriteData(tekst1[i]);

LCD\_WriteCommand( LCD\_DDRAM\_SET | 0x08 );

write\_temp(temp);

temp = (-1481.96 + sqrt( 2196200 + ( ((1.8663 - (wartoscADC1\*0.8067/1000)) )/0.00000388 ) ))\*100;

LCD\_WriteCommand( LCD\_DDRAM\_SET | 0x28 );

for (i=0; i<10 ;i++)

LCD\_WriteData(tekst2[i]);

write\_temp(temp);

for (i=0; i< 3200000ul;i++)

};

while (1);

return 0;

}

void write\_temp(int temp) {

LCD\_WriteData(48 + ((temp/1000)%10));

LCD\_WriteData(48 + (temp/100) % 10);

LCD\_WriteData(',');

LCD\_WriteData(48 + (temp/10)%10);

LCD\_WriteData(48 + temp%10);

LCD\_WriteData(0xdf); LCD\_WriteData('C');

}

void RCC\_Config(void)

//konfigurowanie sygnalow taktujacych

{

ErrorStatus HSEStartUpStatus; //zmienna opisujaca rezultat uruchomienia

HSE RCC\_DeInit(); //Reset ustawien RCC

RCC\_HSEConfig(RCC\_HSE\_ON); //Wlaczenie HSE

HSEStartUpStatus = RCC\_WaitForHSEStartUp(); //Odczekaj az HSE bedzie gotowy

if(HSEStartUpStatus == SUCCESS) {

FLASH\_PrefetchBufferCmd(FLASH\_PrefetchBuffer\_Enable); FLASH\_SetLatency(FLASH\_Latency\_2); //ustaw zwloke dla pamieci Flash; zaleznie od taktowania rdzenia //0:48MHz

RCC\_HCLKConfig(RCC\_SYSCLK\_Div1); //ustaw HCLK=SYSCLK RCC\_PCLK2Config(RCC\_HCLK\_Div1); //ustaw PCLK2=HCLK RCC\_PCLK1Config(RCC\_HCLK\_Div2); //ustaw PCLK1=HCLK/2

RCC\_PLLConfig(RCC\_PLLSource\_HSE\_Div1, RCC\_PLLMul\_9); //ustaw PLLCLK = HSE\*9 czyli 8MHz \* 9 = 72 MHz

RCC\_PLLCmd(ENABLE); //wlacz PLL

while(RCC\_GetFlagStatus(RCC\_FLAG\_PLLRDY) == RESET); //odczekaj na poprawne uruchomienie PLL

RCC\_SYSCLKConfig(RCC\_SYSCLKSource\_PLLCLK); //ustaw PLL jako zrodlo sygnalu zegarowego

while(RCC\_GetSYSCLKSource() != 0x08); //odczekaj az PLL bedzie sygnalem zegarowym systemu

RCC\_ADCCLKConfig(RCC\_PCLK2\_Div6); // ADCCLK = PCLK2/6 = 72 MHz /6 = 12 MHz

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA, ENABLE);//wlacz taktowanie portu GPIO A

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB, ENABLE);//wlacz taktowanie portu GPIO B

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_AFIO, ENABLE));//wlacz taktowanie portu AFIO

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_ADC1, ENABLE); //wlacz taktowanie ADC1

} else { }

}

void GPIO\_Config(void) {

//konfigurowanie portow GPIO

GPIO\_InitTypeDef GPIO\_InitStructure;

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_0|GPIO\_Pin\_1;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AIN;

GPIO\_Init(GPIOA, &GPIO\_InitStructure); // disable JTAG

GPIO\_PinRemapConfig(GPIO\_Remap\_SWJ\_JTAGDisable, ENABLE);

}

int adc\_read(int channel) {

ADC\_RegularChannelConfig(ADC1, channel, 1, ADC\_SampleTime\_239Cycles5); ADC\_SoftwareStartConvCmd(ADC1, ENABLE);

while (!ADC\_GetFlagStatus(ADC1, ADC\_FLAG\_EOC));

return ADC\_GetConversionValue(ADC1);

}

void ADC\_Config(void) {

//konfigurowanie przetwornika AC

ADC\_InitTypeDef ADC\_InitStructure;

ADC\_InitStructure.ADC\_Mode = ADC\_Mode\_Independent; //Jeden przetwornik, praca niezalezna

ADC\_InitStructure.ADC\_ScanConvMode = DISABLE; //Pomiar jednego kanalu, skanowanie kanalow nie potrzebne

ADC\_InitStructure.ADC\_ContinuousConvMode = DISABLE; //Pomiar w trybie jednokrotnym

ADC\_InitStructure.ADC\_ExternalTrigConv = ADC\_ExternalTrigConv\_None; //Brak wyzwalania zewnetrznego

ADC\_InitStructure.ADC\_DataAlign = ADC\_DataAlign\_Right; //Wyrownanie danych do prawej - 12 mlodszych bitow znaczacych

ADC\_InitStructure.ADC\_NbrOfChannel = 1; //Liczba uzywanych kanalow =1

ADC\_Init(ADC1, &ADC\_InitStructure); //Incjalizacja przetwornika

ADC\_RegularChannelConfig(ADC1, ADC\_Channel\_16, 1, ADC\_SampleTime\_239Cycles5); //Kanal 16 - czujnik temperatury procesor

ADC\_Cmd(ADC1, ENABLE); //Wlacz ADC1

ADC\_ResetCalibration(ADC1); //Reset rejestrow kalibracyjnych ADC1 while(ADC\_GetResetCalibrationStatus(ADC1)); //Odczekanie na wykonanie resetu ADC\_StartCalibration(ADC1); //Kalibracja ADC1

while(ADC\_GetCalibrationStatus(ADC1)); //Odczekanie na zakonczenie kalibracji ADC1

}