LAMPIRAN 3

LISTING PROGRAM

Listing Program *Self-Tuning* PID *Controller*

```
#include "stm32f4xx.h"
#include "stm32f4xx_tim.h"
#include "stm32f4xx_gpio.h"
#include "stm32f4xx_rcc.h"
#include "stm32f4xx_adc.h"
#include "stm32f4xx_syscfg.h"
#include "stm32f4xx_usart.h"
#include "misc.h"
#include  "TEUB_USART.h"
#include "TEUB_GPIO.h"

#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdarg.h>

/***Pin IO***/
#define P_0  GPIO_Pin_0
#define P_1  GPIO_Pin_1
#define P_2  GPIO_Pin_2
#define P_3  GPIO_Pin_3
#define P_4  GPIO_Pin_4
#define P_5  GPIO_Pin_5
#define P_6  GPIO_Pin_6
#define P_7  GPIO_Pin_7
#define P_8  GPIO_Pin_8
#define P_9  GPIO_Pin_9
#define P_10  GPIO_Pin_10
```

```
#define P_11  GPIO_Pin_11
#define P_12  GPIO_Pin_12
#define P_13  GPIO_Pin_13
#define P_14  GPIO_Pin_14
#define P_15  GPIO_Pin_15

/***Parameter awal***/
int Q = 1;        // B = 1e-6
int T_0 = 1;     //1 sekon
#define KP 0
#define KI 0
#define KD 0
#define SP 3.2

/***PWM PRESCALER***/
#define   PWM_TIM5_PERIODE    250                    // periode   (100 dutycycle)//1000
#define   PWM_TIM5_PRESCALE   420                    // prescaler (839 => 1kHz)//420
#define   PWM_TIM5_POLARITY   TIM_OCPolarity_High

/***PWM CCR***/
#define   PWM_1R TIM5->CCR1

/***Set variables used***/
GPIO_InitTypeDef              GPIO_InitStructure;
NVIC_InitTypeDef              NVIC_InitStructure;
USART_InitTypeDef             USART_InitStructure;
TIM_TimeBaseInitTypeDef       TIM_TimeBaseStructure;
TIM_OCInitTypeDef         TIM_OCInitStructure;
ADC_InitTypeDef                     ADC_init_structure;
```

```
/***Setting up Variable ADC and USART***/
int ConvertertedValue;
int x=0;
int cnt_usart = 0;
float nilai_pwm = 0;
int cnt = 0;
char usart_sign = 'f';
char str_usart[5];

/***Setting up Variable Dahlin***/
float  Kp = 0;
float Ti = 0;
float Td = 0;

/***Setting up Variable Estimasi RLS***/
float a_hat1, a_hat2, b_hat1;  //komponen dari theta (estimasi parameter)
float y_1, y_2, u_1;    //komponen dari vegtor regresi
float theta_update1 = 0;
float theta_update2 = 0;
float theta_update3 = 0;
float gain_update_11 = 0;
float gain_update_12 = 0;
float gain_update_13 = 0;
float gain_update_21 = 0;
float gain_update_22 = 0;
float gain_update_23 = 0;
float gain_update_31 = 0;
float gain_update_32 = 0;
float gain_update_33 = 0;
```

```
//matrik kovarian F(t) --> C
float gain_adaptasi_11, gain_adaptasi_12, gain_adaptasi_13, gain_adaptasi_21,
gain_adaptasi_22, gain_adaptasi_23, gain_adaptasi_31,
gain_adaptasi_32, gain_adaptasi_33;

/***Setting up Variable PID***/
float out_suhu = 0;
float error = 0;
float error_1 = 0;
float error_2 = 0;
float action = 0;
float action_1 = 0;
float Ki = 0;
float Kd = 0;
float setPoint = 0;
float PID_result = 0;
float out=0;

/* function General */
void setting_ADC();
int  adc_convert();
void TIM5_init();
void setting_GPIO_Driver();
void set_NVIC_USART();
void change_param();
void print_action();
void setParameterAwal();
void read_SuhuAndError();
void action_condition();
void set_param_awal_1();
```

```
//void action_condition();
int fac_x;

/* function PID */
float PID_biasa();
void PID_calculate();
void outkon();
float action_2 = 0;
void change_param_biasa();
float out_1 = 0;
void outkon_1();

/* function Self Tuning Controller */
void RLS ();
void dahlin ();
void update_theta();
void update_inout();

int main(void)
{
	init_USART(RCC_APB1Periph_USART2, 9600, RCC_AHB1Periph_GPIOD,
P_5|P_6, TX_RX);
	Delayms(100);
	set_NVIC_USART();
	setting_GPIO_Driver();
	setting_ADC();
	TIM5_init();
	setParameterAwal();

	read_SuhuAndError();
	y_2 = y_1;
```

```
    y_1 = out_suhu;
    u_1 = PID_result;
    y_2 = y_1;
    y_1 = out_suhu;

while(1)
{
    if (usart_sign == 'a'){
        read_SuhuAndError();
        dahlin();
        PID_calculate();
        action = PID_result;
        outkon();
        PWM_1R = nilai_pwm;
        print_action();
        RLS();
        update_theta();
        Delayms(100);
        update_inout();
        change_param();
        cnt++;
        }

    else if(usart_sign == 'b'){       //nguji PWM dari driver
        PWM_1R = 100;
        read_SuhuAndError();
        print_action();
        Delayms(100);
    }
```

```
else if (usart_sign == 'd'){
  setPoint = 3.6;
  read_SuhuAndError();
  RLS();
  update_theta();
  dahlin();
  PID_calculate();
  action = PID_result;
  outkon();
  PWM_1R = nilai_pwm;
  print_action();
  Delayms(100);
  update_inout();
  change_param();
  cnt++;
   }
else if (usart_sign == 'e'){
      PWM_1R = 0;
      setParameterAwal();
      cnt = 0;
}
else if (usart_sign == 'g'){
  setPoint = 4.0;
  read_SuhuAndError();
  RLS();
  update_theta();
  dahlin();
  PID_calculate();
  action = PID_result;
  outkon();
```

```
            PWM_1R = nilai_pwm;
            print_action();
            Delayms(100);
            update_inout();
            change_param();
            cnt++;
             }
    }
}

/***General Function***/
void setting_ADC(){
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1,ENABLE);
        RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);
        //RCC_AHB1PeriphClockCmd(RCC_AHB1ENR_GPIOAEN, ENABLE);

        GPIO_InitStructure.GPIO_Pin = P_1;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AN;
        GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
        GPIO_Init(GPIOA, &GPIO_InitStructure);

        ADC_DeInit();
        ADC_init_structure.ADC_DataAlign = ADC_DataAlign_Right;
        ADC_init_structure.ADC_Resolution = ADC_Resolution_12b;
        ADC_init_structure.ADC_ContinuousConvMode = ENABLE;
        ADC_init_structure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_T1_CC1;
        ADC_init_structure.ADC_ExternalTrigConvEdge =
ADC_ExternalTrigConvEdge_None;
        ADC_init_structure.ADC_NbrOfConversion = 1;
        ADC_init_structure.ADC_ScanConvMode = DISABLE;
        ADC_Init(ADC1,&ADC_init_structure);
```

```
	ADC_RegularChannelConfig(ADC1,ADC_Channel_1,1,ADC_SampleTime_480Cycles);
	//ADC_EOCOnEachRegularChannelCmd(ADC1,ENABLE);
	ADC_Cmd(ADC1,ENABLE);
}
int adc_convert(){
	ADC_SoftwareStartConv(ADC1);//Start the conversion
	while(!ADC_GetFlagStatus(ADC1, ADC_FLAG_EOC));//Processing the conversion
	return ADC_GetConversionValue(ADC1); //Return the converted data
}
void TIM5_init()
{
	// Clock enable
	RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);	// Clocking GPIOC (AHB1/APB1 = 42MHz)
	RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM5, ENABLE);	// Clocking TIM3  (APB1 = 42x2PLL=84MHz)

	// GPIO__TIM_init
	GPIO_InitStructure.GPIO_Pin   = P_0;			// Ch.1 (PC6), Ch.2 (PC7)
	GPIO_InitStructure.GPIO_Mode  = GPIO_Mode_AF;		// PWM is an alternative function
	GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;		// GPIO_HIGH_Speed
	GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;		// Push-pull
	GPIO_InitStructure.GPIO_PuPd  = GPIO_PuPd_NOPULL;
	GPIO_Init(GPIOA, &GPIO_InitStructure);			// Initializing GPIOC structure
	GPIO_PinAFConfig(GPIOA, GPIO_PinSource0, GPIO_AF_TIM5);	// Routing TIM3 output to PC6
```

```
	// Timer init
	TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;		// => 0 = Not dividing
	TIM_TimeBaseStructure.TIM_CounterMode	= TIM_CounterMode_Up;	// Upcounting configuration
	TIM_TimeBaseStructure.TIM_Period	= PWM_TIM5_PERIODE;	// Autoreload value (ARR)
	TIM_TimeBaseStructure.TIM_Prescaler	= PWM_TIM5_PRESCALE;	// Pembagi Timclock, prescale=>328, Timclock=>84MHz ====> 84Mhz/255/328= mendekati 1kHz
	TIM_TimeBaseInit(TIM5, &TIM_TimeBaseStructure);		// Initializing Time Base structure

	// Channel 1, Ch.1 (PC6)
	TIM_OCInitStructure.TIM_OCMode	= TIM_OCMode_PWM1;		// PWM mode 1 = Set on compare match, PWM mode 2 = Clear on compare match pasangan OCPolarity_LOW
	TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;	// Enabling the Output Compare state
	TIM_OCInitStructure.TIM_OCPolarity = PWM_TIM5_POLARITY;		// Regular polarity (low will inverse it)
	//if (PWM_R>PWM_TIM3_PERIODE) {PWM_R=PWM_TIM3_PERIODE;}
	TIM_OCInitStructure.TIM_Pulse	= PWM_1R;			// Output Compare 1 reg value>>>diberi nilai PWM
	TIM_OC1Init(TIM5, &TIM_OCInitStructure);		// Initializing Output Compare 1 structure
	TIM_OC1PreloadConfig(TIM5, TIM_OCPreload_Enable);		// Enable Ch.1 Output Compare preload

	// Timer enable
	TIM_Cmd(TIM5, ENABLE);
}
void setting_GPIO_Driver()
{
	RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD,ENABLE);
```

```
	GPIO_InitStructure.GPIO_Pin = P_1 | P_2;
	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
	GPIO_InitStructure.GPIO_Speed = GPIO_Speed_25MHz;
	GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
	GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
	GPIO_Init(GPIOD, &GPIO_InitStructure);
	GPIO_SetBits(GPIOD, P_2);
	GPIO_ResetBits(GPIOD, P_1);
}
void set_NVIC_USART()
{
  NVIC_InitStructure.NVIC_IRQChannel = USART2_IRQn;      // we want to configure the USART2 interrupts
  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;// this sets the priority group of the USART2 interrupts
  NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;     // this sets the subpriority inside the group
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;        // the USART1 interrupts are globally enabled
  NVIC_Init(&NVIC_InitStructure);                        // the properties are passed to the NVIC_Init function which takes care of the low level stuff
}

void change_param(){
	u_1 = PID_result;
	error_2 = error_1;
	error_1 = error;
	action_1 = nilai_pwm;
	y_2 = y_1;
	y_1 = out_suhu;
	gain_adaptasi_11 = gain_update_11;
	gain_adaptasi_12 = gain_update_12;
```

```
        gain_adaptasi_13 = gain_update_13;
        gain_adaptasi_21 = gain_update_21;
        gain_adaptasi_22 = gain_update_22;
        gain_adaptasi_23 = gain_update_23;
        gain_adaptasi_31 = gain_update_31;
        gain_adaptasi_32 = gain_update_32;
        gain_adaptasi_33 = gain_update_33;
}

void update_inout(){
        float Z = 0;
        Z = action;
        if (Z > 15){//15
                PID_result = 15;
        }
        else {
                PID_result = Z;
        }
}
void update_theta(){
        a_hat1 = theta_update1;
        a_hat2 = theta_update2;
        b_hat1 = theta_update3;
}
void print_action(){
        if(usart_sign=='a'){
                        cetak(USART2,"%d\t %.3f\t %.3f\t %f\t %f\t %f\t %f\t %f\t %f\t
%f\n", cnt, setPoint, out_suhu,a_hat1, a_hat2, b_hat1, Kp, Ki, Kd,PID_result);
                }
        else if(usart_sign == 'b'){
                cetak(USART2,"mode PWM & ADC = %.3f\n", out_suhu);
```

```
        }
        else if(usart_sign=='d'){
                cetak(USART2,"%d\t %.3f\t %.3f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\n",
cnt, setPoint, out_suhu,a_hat1, a_hat2, b_hat1, Kp, Ki, Kd,PID_result);
                        }
        else if(usart_sign=='g'){
                cetak(USART2,"%d\t %.3f\t %.3f\t %f\t %f\t %f\t %f\t %f\t %f\t %f\n",
cnt, setPoint, out_suhu,a_hat1, a_hat2, b_hat1, Kp, Ki, Kd,PID_result);
        }
        else{}
}
void setParameterAwal(){
        Kp = KP;
        Ki = KI;
        Kd = KD;
        setPoint = SP;
        y_1 = y_2 = u_1 = 0;
        Ti = Td = 0;
        a_hat1 = -0.949214468;
        a_hat2 = 0.272195834;
        b_hat1 = 0.007307270736;
        theta_update1 = 0;
        theta_update2 = 0;
        theta_update3 = 0;

        gain_adaptasi_11 = 10;
        gain_adaptasi_12 = 0;
        gain_adaptasi_13 = 0;
        gain_adaptasi_21 = 0;
        gain_adaptasi_22 = 10;
        gain_adaptasi_23 = 0;
```

```
gain_adaptasi_31 = 0;
        gain_adaptasi_32 = 0;
        gain_adaptasi_33 = 10;
}
void read_SuhuAndError(){
        ConvertertedValue = adc_convert();
        out_suhu = ConvertertedValue*0.00144;
        error = setPoint - out_suhu;
}

void USART2_IRQHandler()
{
        if( USART_GetITStatus(USART2, USART_IT_RXNE) ){
                char x = USART2->DR;
                if(x == 'a'){usart_sign = 'a';}
                else if(x == 'b'){usart_sign = 'b';}
                else if(x == 'd'){usart_sign = 'd';}
                else if(x == 'e'){usart_sign = 'e';}
                else if(x == 'g'){usart_sign = 'g';}
                else{usart_sign = 'f';}
        }
}

/*** Setting Output Kontroller ***/
void outkon (){
        if (action < 0){
                nilai_pwm = 0;
        }
        else if (action > 250){
                nilai_pwm = 250;
```

```
        }
        else {
                nilai_pwm = action;
        }
}

/*** Desain Control PID Dahlin ***/
void dahlin(){
        float F, G, M, N, H, J, L;

        F = a_hat1+(2*a_hat2);
        G = (F/b_hat1);
        Kp = -1*(G*Q);          //nilai Kp

        M = T_0*a_hat2*Q;
        N = Kp*b_hat1;
        Td = (M/N);    //nilai Td

        H = (1/F);
        J = (Td/T_0);
        L = H+1+J;
        Ti = -1*(T_0/L);        //nilai Ti

        Ki = (Kp/Ti);
        Kd = Kp*Td;
}

/*** PID Function ***/
void PID_calculate(){
          float q0, q1, q2, p1, p2;
```

```
        q0 = Kp*(1 + (T_0/Ti) + (Td/T_0));
        q1 = -Kp*(1 + 2*(Td/T_0));
        q2 = Kp*(Td/T_0);
        p1 = -1;
        p2 = 0;

        PID_result = (q0*error) + (q1*error_1) + (q2*error_2) - (p1*action_1);
}

/*** RLS (Recursive Least Square) ***/
void RLS (){
        float out_estimasi, error_estimasi, aux_cal1, aux_cal2, aux_cal3, aux_param,
aux_theta1, aux_theta2, aux_theta3,
        denom;

        float aux_11a, aux_21a, aux_31a, aux_11b, aux_12b, aux_13b, aux_21b, aux_22b,
aux_23b, aux_31b, aux_32b, aux_33b,
        aux_11c, aux_12c, aux_13c, aux_21c, aux_22c, aux_23c, aux_31c, aux_32c,
aux_33c;

        out_estimasi = (a_hat1*-1*y_1) + (a_hat2*-1*y_2) + (b_hat1*u_1);
        error_estimasi = out_suhu - out_estimasi;

        aux_cal1 = (-1*y_1*gain_adaptasi_11) + (-1*y_2*gain_adaptasi_21)
+(u_1*gain_adaptasi_31);
        aux_cal2 = (-1*y_1*gain_adaptasi_12) + (-1*y_2*gain_adaptasi_22)
+(u_1*gain_adaptasi_32);
        aux_cal3 = (-1*y_1*gain_adaptasi_13) + (-1*y_2*gain_adaptasi_23)
+(u_1*gain_adaptasi_33);
        aux_param = (aux_cal1*-1*y_1) + (aux_cal2*-1*y_2) + (aux_cal3*u_1);

        denom = 1 + aux_param;
```

```
// theta update (update estimasi parameter)
aux_theta1 = (((gain_adaptasi_11*-1*y_1) + (gain_adaptasi_12*-1*y_2) +
(gain_adaptasi_13*u_1))*error_estimasi)/denom;
aux_theta2 = (((gain_adaptasi_21*-1*y_1) + (gain_adaptasi_22*-1*y_2) +
(gain_adaptasi_23*u_1))*error_estimasi)/denom;
aux_theta3 = (((gain_adaptasi_31*-1*y_1) + (gain_adaptasi_32*-1*y_2) +
(gain_adaptasi_33*u_1))*error_estimasi)/denom;
theta_update1 = a_hat1 + aux_theta1;
theta_update2 = a_hat2 + aux_theta2;
theta_update3 = b_hat1 + aux_theta3;

//update gain adaptasi (matrik kovarian)
aux_11a = (gain_adaptasi_11*-1*y_1) + (gain_adaptasi_12*-1*y_2) +
(gain_adaptasi_13*u_1);
aux_21a = (gain_adaptasi_21*-1*y_1) + (gain_adaptasi_22*-1*y_2) +
(gain_adaptasi_23*u_1);
aux_31a = (gain_adaptasi_31*-1*y_1) + (gain_adaptasi_32*-1*y_2) +
(gain_adaptasi_33*u_1);

aux_11b = (aux_11a*-1*y_1);
aux_12b = (aux_11a*-1*y_2);
aux_13b = (aux_11a*u_1);

aux_21b = (aux_21a*-1*y_1);
aux_22b = (aux_21a*-1*y_2);
aux_23b = (aux_21a*u_1);

aux_31b = (aux_31a*-1*y_1);
aux_32b = (aux_31a*-1*y_2);
aux_33b = (aux_31a*u_1);
```

```
        aux_11c = ((aux_11b*gain_adaptasi_11) + (aux_12b*gain_adaptasi_21) +
(aux_13b*gain_adaptasi_31))/denom;

        aux_12c = ((aux_11b*gain_adaptasi_12) + (aux_12b*gain_adaptasi_22) +
(aux_13b*gain_adaptasi_32))/denom;

        aux_13c = ((aux_11b*gain_adaptasi_13) + (aux_12b*gain_adaptasi_23) +
(aux_13b*gain_adaptasi_33))/denom;


        aux_21c = ((aux_21b*gain_adaptasi_11) + (aux_22b*gain_adaptasi_21) +
(aux_23b*gain_adaptasi_31))/denom;

        aux_22c = ((aux_21b*gain_adaptasi_12) + (aux_22b*gain_adaptasi_22) +
(aux_23b*gain_adaptasi_32))/denom;

        aux_23c = ((aux_21b*gain_adaptasi_13) + (aux_22b*gain_adaptasi_23) +
(aux_23b*gain_adaptasi_33))/denom;


        aux_31c = ((aux_31b*gain_adaptasi_11) + (aux_32b*gain_adaptasi_21) +
(aux_33b*gain_adaptasi_31))/denom;

        aux_32c = ((aux_31b*gain_adaptasi_12) + (aux_32b*gain_adaptasi_22) +
(aux_33b*gain_adaptasi_32))/denom;

        aux_33c = ((aux_31b*gain_adaptasi_13) + (aux_32b*gain_adaptasi_23) +
(aux_33b*gain_adaptasi_33))/denom;


        gain_update_11 = (gain_adaptasi_11 - aux_11c);

        gain_update_12 = (gain_adaptasi_12 - aux_12c);

        gain_update_13 = (gain_adaptasi_13 - aux_13c);


        gain_update_21 = (gain_adaptasi_21 - aux_21c);

        gain_update_22 = (gain_adaptasi_22 - aux_22c);

        gain_update_23 = (gain_adaptasi_23 - aux_23c);


        gain_update_31 = (gain_adaptasi_31 - aux_31c);

        gain_update_32 = (gain_adaptasi_32 - aux_32c);

        gain_update_33 = (gain_adaptasi_33 - aux_33c);

}
```